Document Title	SWS_LargeDataCOM: Complete Change Documentation 4.3.0 - 4.3.1
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	695

Document Status	Final
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	4.3.1

Table of Contents

1 SWS_LargeDataCOM

1.1 Specification Item SWS_LDCOM_00009

Trace References:

SRS_Com_02108

Content:

When called by RTE the RTE, LdCom shall use the Signal Id as Handle ("id" parameter in the call) as LdComHandleId (ECUC_LdCom_00005) , passed as Parameter to derive the according PDU (to look-up the correct LdComIPdu in the LdCom configuration. Using the LdComPduRef configuration parameter (ECUC_LdCom_00010) when passing the corresponding PDU Id in the PduR'S configuration shall be derived. This PDU Id shall then be used when forwarding the call towards the PduR.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #76273: Clarification regrading "signal ID" in SWS_LDCOM_00009

    Problem description:

    At the time being the term "signal ID" in SWS_LDCOM_00009 is misleading and would benefit from clarification how this is related to the configuration parameter LdComHandleId of the LdComIPdu config container.

    Agreed solution:

    [SWS_LDCOM_00009] When called by the RTE LdCom shall use the Signal Id ("id" parameter in the call) as
    LdComHandleId (ECUC_LdCom_00005) to look-up the correct LdComIPdu in the LdCom configuration. Using the LdComPduRef configuration parameter (ECUC_LdCom_00010) the corresponding PDU Id in the PduR'S configuration shall be derived. This PDU id shall then be used when forwarding the call towards the PduR. (SRS_Com_02108)
    –Last change on issue 76273 comment 7–

    BW-C-Level:

Application	Specification	Bus
1	1	1

Document ID 695: ChangeDocumentation

1.2 Specification Item SWS_LDCOM_00026

Trace References:

SRS_Com_02110

Content:

Service name:	LdCom_TransmitLdCom_Transmit	
Syntax:	void Std_ReturnType LdCom_Transmit(PduIdType Id, const PduInfoType* PduInfoPtr)	
Service ID[hex]:	0x05	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant for same HandleId, otherwise Reentrant Reentrant for different Ids. Non reentrant for the same Id.	
Parameters (in):	IdLdCom_Transmit.Id	Id Identifier of the signal to be sent transmitted.
	PduInfoPtrLdCom_Transmit.InfoPtr	Length of and pointer to the buffer of the Signalsignal data and pointer to Meta Data.
Parameters (inout):	None	
Parameters (out):	None	

None

Return value:	Std_ReturnType	E_OK: Transmit request has been accepted. E_NOT_OK: Transmit request has not been accepted.
Description:	Initiate a Requests transmission of a signalusing either IF- or TP-API set.	

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #76470: [LdCom] Return type of LdCom_Tansmit shall be Std_ReturnType

  Problem description:

  ───────────────────────────

  Description/Motivation:
  For some unknown reason LdCom_Transmit has become void in Rel. 4.3.
  But RTE has to evaluate the return value (SWS_Rte_01379).

  ───────────────────────────

  Was there already a decision?

  ───────────────────────────

Agreed solution:

=== SWS LdCom / BSW UML ===
- Change description of LdCom_Transmit to: "Requests transmission of a signal."
- Change reentrancy of LdCom_Transmit to: "Reentrant for different Ids. Non reentrant for the same Id."
- Change description of parameter Id of LdCom_Transmit to: "Identifier of the signal to be transmitted."
- Change name of parameter PduInfoPtr to InfoPtr and description to: "Length of and pointer to the signal data and pointer to MetaData."
- Add return value Std_ReturnType to LdCom_Transmit, description:
"E_OK: Transmit request has been accepted.
E_NOT_OK: Transmit request has not been accepted."
–Last change on issue 76470 comment 8–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.3 Specification Item SWS_LDCOM_00027

Trace References:

SRS_Com_02109

Content:

Service name:	LdCom_CopyTxDataLdCom_CopyTxData
Syntax:	BufReq_ReturnType LdCom_CopyTxData(PduIdType id, const PduInfoType* info, const RetryInfoType* retry, PduLengthType* availableDataPtr)
Service ID[hex]:	0x43
Sync/Async:	Synchronous
Reentrancy:	Reentrant

Parameters (in):	idLdCom_CopyTxData.id	Identification of the transmitted I-PDU.
	infoLdCom_CopyTxData.info	Provides the destination buffer (SduData Ptr) and the number of bytes to be copied (SduLength). If not enough transmit data is available, no data is copied by the upper layer module and BUFREQ_E_BUSY is returned. The lower layer module may retry the call. An SduLength of 0 can be used to indicate state changes in the retry parameter or to query the current amount of available data in the upper layer module. In this case, the SduDataPtr may be a NULL_PTR.
	retryLdCom_CopyTxData.retry	This parameter is used to acknowledge transmitted data or to retransmit data after transmission problems. If the retry parameter is a NULL_PTR, it indicates that the transmit data can be removed from the buffer immediately after it has been copied. Otherwise, the retry parameter must point to a valid RetryInfo Type 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 has been copied before this call is confirmed and can be removed from the TP buffer. Data copied by this API call is excluded and will be confirmed later. TP_DATARETRY indicates that this API call shall copy previously copied data in order to recover from an error. In this case TxTpDataCnt specifies the offset in bytes from the current data copy position.
Parameters (inout):	None	
Parameters (out):	availableDataPtrLdCom_CopyTx Data.availableDataPtr	Indicates the remaining number of bytes that are available in the upper layer module's Tx buffer. availableDataPtr can be used by TP modules that support dynamic payload lengths (e.g. FrIsoTp) to determine the size of the following CFs.
Return value:	BufReq_ReturnType	BUFREQ_OK: Data has been copied to the transmit buffer completely as requested. BUFREQ_E_BUSY: Request could not be fulfilled, because the required amount of Tx data is not available. The lower layer module may retry this call later on. No data has been copied. BUFREQ_E_NOT_OK: Data has not been copied. Request failed.
Description:	This function is called to acquire the transmit data of an I-PDU segment (N-PDU). Each call to this function provides the next part of the I-PDU data unless retry->Tp DataState is TP_DATARETRY. In this case the function restarts to copy the data beginning at the offset from the current position indicated by retry->TxTpDataCnt. The size of the remaining data is written to the position indicated by availableDataPtr.	

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #68035: [diverse] Introduce rules defining which input parameters shall be passed per value and which ones per const reference

Problem description:

SWS_BSW_00186 especially states that input pointer parameters shall use the const qualifier (i.e., shall be P2CONST).

In addition to that there shall be a SWS item that states that input parameters of integral and enum type shall be passed by value whereas input parameters of structure type shall be passed by reference.

The various transformer SWS documents shall be adapted accordingly.
–Last change on issue 68035 comment 4–

Agreed solution:

BSW UML model
————-
The attachment "Changed Proposal in WP-A meeting" contains a list of changes to the APIs in the model (see column H). Afterwards all related documents (included in impact list) shall update their
generated artifacts.

General Requirements on Basic Software Modules
~~~~~~

Introduce the following requirements prior to SRS_BSW_00371:

SRS_BSW_xxxxx: Input parameters of scalar and enum types shall be passed as a value.
Type: valid
Description: All input parameters of scalar or enum type shall be passed as a value.
Rationale:

Use case: For example a function named <Mip>_SomeFunction with a return type of Std_ReturnType and a single parameter named SomeParameter of type uint8 is defined with the following signature:

Std_ReturnType <Mip>_SomeFunction(uint8 SomeParameter);
Dependencies: –

Supporting Material: —

SRS_BSW_yyyyy: Input parameters of structure type shall be passed as a reference to a constant structure

Type: valid

Description: All input parameters of structure type shall be passed as a reference constant structure

Rationale: Passing input parameters of structure type by value would result in additional run-time overhead due to efforts for copying the whole structure.

Use case: For example a function named <Mip>_SomeFunction with a return type of Std_ReturnType and a single parameter named SomeParameter of type SomeStructure (where SomeStructure is a struct) is defined with the following signature:

Std_ReturnType <Mip>_SomeFunction(P2CONST(SomeStructure, AUTOMATIC, <MIP>_APPL_DATA) SomeParameter);

Dependencies: –

Supporting Material: —

SRS_BSW_zzzzz: Input parameters of array type shall be passed as a reference to the constant array base type

Type: valid

Description: All input parameters of array type shall be passed as a reference to the constant array base type

Rationale: This effectively matches the behavior specified in the ISO-C:90 namely that a "declaration of a parameter as 'array of type' shall be adjusted to 'qualified pointer to

type'".

Use case: For example a function named <Mip>_SomeFunction with a return type of Std_ReturnType and a single parameter named SomeParameter of type array of uint8 is defined with the following signature:

Std_ReturnType <Mip>_SomeFunction(P2CONST(uint8, AUTOMATIC, <MIP>_APPL_DATA) SomeParameter);

Dependencies: –

Supporting Material: —

General Specification of Transformers
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

In SWS_Xfrm_00036 change

const <type>* dataElement

to

<paramtype> dataElement

and add the following to the where clause after the API table after the bullet "type is data type of the data element
"

<paramtype> is derived from <type> according to the parameter passing rules rules defined by the SRS BSW General (see SRS_BSW_xxxxx, SRS_BSW_yyyyy, and SRS_BSW_zzzzz) and SWS BSW General (see SWS_BSW_00186 and SWS_BSW_00187).

In SWS_Xfrm_00038 change

[<type> data_1,] ...
[<type> data_n]

to

[<paramtype> data_1,] ...
[<paramtype> data_n]

and add the following to the where clause after the API table after the bullet "type is data type of the data element
"

<paramtype> is derived from <type> according to the parameter passing rules rules defined by the SRS BSW General (see SRS_BSW_xxxxx, SRS_BSW_yyyyy, and SRS_BSW_zzzzz) and SWS BSW General (see SWS_BSW_00186 and SWS_BSW_00187).

The following paragraph shall then be removed:

For the arguments of ClientServerOperation which are handed over to the transformer as data_1, ..., data_n the requirements to API parameters stated in chapter API Parameters of [5, SWS RTE] are valid (especially [SWS_Rte_01017],

[SWS_Rte_01018] and [SWS_Rte_05107]).

In SWS_Xfrm_00040 change

[<originalData1>, ...
<originalDataN>]

to

[<paramtype> originalData1,] ...
[<paramtype> originalDataN]

and add the following to the where clause after the API table after the bullet
"type is data type of the data element
"

<paramtype> is derived from <type> according to the parameter passing rules
rules defined by the SRS BSW General (see SRS_BSW_xxxxx, SRS_BSW_yyyyy,
and SRS_BSW_zzzzz) and SWS BSW General (see SWS_BSW_00186 and
SWS_BSW_00187).

In SWS_Xfrm_00044 change

<type> *data_1, ...
<type> *data_n

to

[<paramtype> data_1,] ...
[<paramtype> data_n]

and add the following to the where clause after the API table after the bullet
"type is data type of the data element
"

<paramtype> is derived from <type> according to the parameter passing rules
rules defined by the SRS BSW General (see SRS_BSW_xxxxx, SRS_BSW_yyyyy,
and SRS_BSW_zzzzz) and SWS BSW General (see SWS_BSW_00186 and
SWS_BSW_00187).

The following paragraph shall then be removed:

For the arguments of ClientServerOperation which are handed over to the transformer as data_1, ..., data_n the requirements to API parameters stated in chapter API Parameters of [5, SWS RTE] are valid (especially [SWS_Rte_01017], [SWS_Rte_01018] and [SWS_Rte_05107]).

Speci?cation of SOME/IP Transformer
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

In SWS_SomeIpXf_00138 change

const <type>* dataElement

to

<paramtype> dataElement

and add the following to the where clause after the API table after the bullet
"type is data type of the data element
"

<paramtype> is derived from <type> according to the parameter passing rules rules defined by the SRS BSW General (see SRS_BSW_xxxxx, SRS_BSW_yyyyy, and SRS_BSW_zzzzz) and SWS BSW General (see SWS_BSW_00186 and SWS_BSW_00187).

In SWS_SomeIpXf_00141 change

[<type> data_1,] ...
[<type> data_n]

to

[<paramtype> data_1,] ...
[<paramtype> data_n]

and add the following to the where clause after the API table after the bullet
"type is data type of the data element
"

<paramtype> is derived from <type> according to the parameter passing rules rules defined by the SRS BSW General (see SRS_BSW_xxxxx, SRS_BSW_yyyyy, and SRS_BSW_zzzzz) and SWS BSW General (see SWS_BSW_00186 and SWS_BSW_00187).

The following paragraph shall then be removed:

For the arguments of ClientServerOperation which are handed over to the transformer as data_1, ..., data_n the requirements to API parameters stated in chapter API Parameters of [5, SWS RTE] are valid (especially [SWS_Rte_01017], [SWS_Rte_01018] and [SWS_Rte_05107]).

In SWS_SomeIpXf_00145 change

<type> *data_1, ...
<type> *data_n

to

[<paramtype> data_1,] ...
[<paramtype> data_n]

and add the following to the where clause after the API table after the bullet "type is data type of the data element
"

<paramtype> is derived from <type> according to the parameter passing rules rules defined by the SRS BSW General (see SRS_BSW_xxxxx, SRS_BSW_yyyyy, and SRS_BSW_zzzzz) and SWS BSW General (see SWS_BSW_00186 and SWS_BSW_00187).

The following paragraph shall then be removed:

For the arguments of ClientServerOperation which are handed over to the transformer as data_1, ..., data_n the requirements to API parameters stated in chapter API Parameters of [5, SWS RTE] are valid (especially [SWS_Rte_01017], [SWS_Rte_01018] and [SWS_Rte_05107]).

Specification of COM Based Transformer

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

In SWS_ComXf_00007 change

const <type>* dataElement

to

<paramtype> dataElement

and add the following to the where clause after the API table after the bullet
"type is data type of the data element
"

<paramtype> is derived from <type> according to the parameter passing rules
rules defined by the SRS BSW General (see SRS_BSW_xxxxx, SRS_BSW_yyyyy,
and SRS_BSW_zzzzz) and SWS BSW General (see SWS_BSW_00186 and
SWS_BSW_00187).

Specification of Time Sync over Ethernet
~~~~~~

In SWS_EthTSyn_00040 make the parameter DataPtr of EthTSyn_RxIndication
const.

Specification of SWS FlexRay Interface
~~~~~~

Change SWS_FrIf_05073 from
FrIf_NumOfStartupFramesPtr (IN)
to
FrIf_NumOfStartupFramesPtr (OUT)

Specification of ADC
~~~~~~
~[SWS_Adc_00419] Adc_SetupResultBuffer: change Adc_ValueGroupType* to
const Adc_ValueGroupType*
~[SWS_Adc_00369] Adc_ReadGroup: move Adc_ValueGroupType * from Parame-

ters (in) to Parameters (out)

There is no need to change parameter from IN to INOUT in Adc_SetupResultBuffer

Specification of Com
~~~~~~
Change type of parameter MetaData of Com_TriggerIPDUSendWithMetaData from
uint8* to const uint8*

Specification of ComM
~~~~~~
no change required

Specification of Dem
~~~~~~
no change required

Specification of DLT
~~~~~~
no change required

Specification of DoIP
~~~~~~
From:
Std_ReturnType <User>_DoIPRoutingActivationConfirmation(boolean* Confirmed,
uint8* ConfirmationReqData, uint8* ConfirmationResData)
Std_ReturnType <User>_DoIPRoutingActivationAuthentication(boolean* Authenti-
fied, uint8* AuthenticationReqData, uint8* AuthenticationResData)

To:
Std_ReturnType <User>_DoIPRoutingActivationConfirmation(boolean* Confirmed,
const uint8* ConfirmationReqData, uint8* ConfirmationResData)
Std_ReturnType <User>_DoIPRoutingActivationAuthentication(boolean* Authenti-
fied, const uint8* AuthenticationReqData, uint8* AuthenticationResData)

Specification of E2ELibrary
~~~~~~

no change required

Specification of Eth
~~~~~~
no change required

Specification of EthIf
~~~~~~
no change required

Specification of EthSwitchDriver
~~~~~~
no change required

Specification of ICUDriver
~~~~~~
SWS_Icu_00201: Icu_StartTimestamp
Parameter (IN): Icu_ValueType* BufferPtr shall be changed to Parameters (out) type

Specification of LdCom
~~~~~~
[SWS_LDCOM_00027]: LdCom_CopyTxData
BufReq_ReturnType LdCom_CopyTxData(PduIdType id, const PduInfoType* info,
RetryInfoType* retry, PduLengthType* availableDataPtr) shall be changed to
BufReq_ReturnType LdCom_CopyTxData(PduIdType id, const PduInfoType* info,
const RetryInfoType* retry, PduLengthType* availableDataPtr)

[SWS_LDCOM_00036]: Rte_LdComCbkCopyTxData_<sn>
BufReq_ReturnType Rte_LdComCbkCopyTxData_<sn>(const PduInfoType* info,
RetryInfoType* retry, PduLengthType* availableDataPtr) shall be changed to
BufReq_ReturnType Rte_LdComCbkCopyTxData_<sn>(const PduInfoType* info,
const RetryInfoType* retry, PduLengthType* availableDataPtr)

Specification of Lin
~~~~~~
PduInfoPtr needs to be const in Std_ReturnType Lin_SendFrame(uint8 Channel,
const Lin_PduType* PduInfoPtr)

Specification of PduR
~~~~~~
* PduR_<User:LoTp>CopyTxData
add const to "RetryInfoType* retry"

Specification of J1939Nm
~~~~~~
Change parameter 'name' of User_AddressClaimedIndication to type 'const uint8*'

Specification of SoAd
~~~~~~
=> everything already fixed with RfC 65633

Specification of SPIHandlerDriver
~~~~~~
==> nothing to change for SWS SPI

Specification of SynchronizedTimeBaseManager
~~~~~~
"StbM not affected. All issues listed in the WP-A attachment have been already
implemented by IT 69124 in context of RfC 65633"

Specification of TcpIp
~~~~~~
~[SWS_TCPIP_00040] TcpIp_DhcpReadOption: change DataPtr from (IN) to
(OUT)
~[SWS_TCPIP_00189] TcpIp_DhcpV6ReadOption: change DataPtr from (IN) to
(OUT)
=> everything else already fixed with RfC 65633

Specification of TimeSyncOverFlexRay
~~~~~~
"Change SWS_FrTSyn_00064: parameter versioninfo of type Std_VersionInfoType*
is marked wrongly as IN. Change to OUT"

Specification of EFX
~~~~~~~~~~~~~~~~~~~~~~
~ [SWS_Efx_00355] Efx_Debounce_u8_u8: Include constant for pointer Input-parameter as like below.
uint8 Efx_Debounce_u8_u8(boolean X, Efx_DebounceState_Type * State, const Efx_DebounceParam_Type * Param, sint32 dT)

~ [SWS_Efx_00376] Efx_MedianSort: The parameter <InType>* Array should be InOut instead of In parameter as like below.
Parameters (in): N Size of an array
Parameters (inout): Array Pointer to an array

~ [SWS_Efx_00309] Efx_RampCheckActivity: Include constant for pointer Input-parameter as like below.
boolean Efx_RampCheckActivity(const Efx_StateRamp_Type* State_cpst)

~ [SWS_Efx_00307] Efx_RampGetSwitchPos: Include constant for pointer Input-parameter as like below.
boolean Efx_RampGetSwitchPos(const Efx_StateRamp_Type* State_cpst)

~ [SWS_Efx_00193] Efx_Array_Average: Include constant for pointer Input-parameter as like below.
<OutType> Efx_Array_Average_<InTypeMn>_<OutTypeMn>(const <InType>* Array, uint16 Count)

Specification of MFL
~~~~~~~~~~~~~~~~~~~~~~
~ [SWS_Mfl_00192] Mfl_Debounce_u8_u8: Include constant for pointer Input-parameter as like below.
boolean Mfl_Debounce_u8_u8(boolean X, Mfl_DebounceState_Type* State, const Mfl_DebounceParam_Type* Param, float32 dT)

~ [SWS_Mfl_00266] Mfl_DebounceInit: The parameter Mfl_DebounceState_Type* State should be Out instead of In parameter as like below.
Parameters (in): X Initial value for the input state
Parameters (out): State Pointer to structure for debouncing state variables

~ [SWS_Mfl_00246] Mfl_HystDeltaRight_f32_u8: Include constant for pointer Input-parameter as like below.
boolean Mfl_HystDeltaRight_f32_u8(float32 X, float32 Delta, float32 Rsp, const uint8* State)

~ [SWS_Mfl_00285] Mfl_MedianSort_f32_f32: The parameter Array should be InOut instead of In parameter as like below.
Parameters (in): N Size of an array
Parameters (inout): Array Pointer to an array

~ [SWS_Mfl_00037] Mfl_PT1SetState: The parameter State_cpst should be Out instead of In parameter as like below.
Parameters (in): X1_f32 Initial value for input state
Y1_f32 Initial value for output state
Parameters (out): State_cpst Pointer to internal state structure

~ [SWS_Mfl_00225] Mfl_RampCheckActivity: Include constant for pointer Input-parameter as like below.
boolean Mfl_RampCheckActivity(const Mfl_StateRamp_Type* State_cpst)

~ [SWS_Mfl_00223] Mfl_RampGetSwitchPos: Include constant for pointer Input-parameter as like below.
boolean Mfl_RampGetSwitchPos(const Mfl_StateRamp_Type* State_cpst)
–Last change on issue 68035 comment 135–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.4   Specification Item SWS_LDCOM_00030

Trace References:

SRS_Com_02109

Content:

Service name:	LdCom_CopyRxDataLdCom_CopyRxData
Syntax:	BufReq_ReturnType LdCom_CopyRxData(PduIdType id, const PduInfoType* info, PduLengthType* bufferSizePtr)
Service ID[hex]:	0x44
Sync/Async:	Synchronous
Reentrancy:	Reentrant

Parameters (in):	idLdCom_CopyRxData.id	Identification of the received I-PDU.
	infoLdCom_CopyRxData.info	Provides the source buffer (SduDataPtr) and the number of bytes to be copied (SduLength). An SduLength of 0 can be used to query the current amount of available buffer in the upper layer module. In this case, the SduDataPtr may be a NULL_PTR.
Parameters (inout):	None	
Parameters (out):	bufferSizePtrLdCom_CopyRxData.bufferSizePtr	Available receive buffer after data has been copied.
Return value:	BufReq_ReturnType	BUFREQ_OK: Data copied successfully BUFREQ_E_NOT_OK: Data was not copied because an error occurred.
Description:	This function is called to provide the received data of an I-PDU segment (N-PDU) to the upper layer. Each call to this function provides the next part of the I-PDU data. The size of the remaining data buffer is written to the position indicated by bufferSizePtr.	

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77935: [PduR] Misleading description of CopyRxData

  Problem description:

  ————————————————
  Name: Martin Schlodder
  Role: Member of WP-A2
  ————————————————

  Description/Motivation:

  The description of the CopyRxData API says: "The size of the remaining data is written to the position indicated by bufferSizePtr."

  This text seems to have been copied from the CopyTxData call, where it is correct. CopyRxData should talk about "remaining buffer", not "remaining data".

  Agreed solution:

  In the description of the API PduR_<User:LoTp>CopyRxData (SWS_PduR_00512), replace "remaining data" by "remaining buffer".

  BW-C-Level:

Application	Specification	Bus
1	1	1

Document ID 695: ChangeDocumentation

1.5 Specification Item SWS_LDCOM_00036

Trace References:

SRS_Com_02109

Content:

Service name:	Rte_LdComCbkCopyTxData_<sn>Rte_LdComCbkCopyTxData_sn
Syntax:	BufReq_ReturnType Rte_LdComCbkCopyTxData_<sn>(const PduInfoType* info, const RetryInfoType* retry, PduLengthType* availableDataPtr)
Sync/Async:	Synchronous
Reentrancy:	Non Reentrant for same sn, otherwise Reentrant

Parameters (in):	infoRte_LdComCbkCopyTxData_sn.info	Provides the destination buffer (SduData Ptr) and the number of bytes to be copied (SduLength). If not enough transmit data is available, no data is copied by the upper layer module and BUFREQ_E_BUSY is returned. The lower layer module may retry the call. An SduLength of 0 can be used to indicate state changes in the retry parameter or to query the current amount of available data in the upper layer module. In this case, the SduDataPtr may be a NULL_PTR.
	retryRte_LdComCbkCopyTx Data_sn.retry	Will not be handled by LdCom and its upper layer.
Parameters (inout):	None	
Parameters (out):	availableDataPtrRte_LdComCbkCopyTx Data_sn.availableDataPtr	Indicates the remaining number of bytes that are available in the upper layer module's Tx buffer. availableDataPtr can be used by TP modules that support dynamic payload lengths (e.g. FrIsoTp) to determine the size of the following CFs.
Return value:	BufReq_ReturnType	BUFREQ_OK: Data has been copied to the transmit buffer completely as requested. BUFREQ_E_BUSY: Request could not be fulfilled, because the required amount of Tx data is not available. The lower layer module may retry this call later on. No data has been copied. BUFREQ_E_NOT_OK: Data has not been copied. Request failed.
Description:	This function is called to acquire the transmit data of an I-PDU segment (N-PDU). Each call to this function provides the next part of the I-PDU data unless retry->Tp DataState is TP_DATARETRY. In this case the function restarts to copy the data beginning at the offset from the current position indicated by retry->TxTpDataCnt. The size of the remaining data is written to the position indicated by availableDataPtr	

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

Document ID 695: ChangeDocumentation

● RfC #68035: [diverse] Introduce rules defining which input parameters shall be passed per value and which ones per const reference

Problem description:

SWS_BSW_00186 especially states that input pointer parameters shall use the const qualifier (i.e., shall be P2CONST).

In addition to that there shall be a SWS item that states that input parameters of integral and enum type shall be passed by value whereas input parameters of structure type shall be passed by reference.

The various transformer SWS documents shall be adapted accordingly.
–Last change on issue 68035 comment 4–

Agreed solution:

BSW UML model
————-
The attachment "Changed Proposal in WP-A meeting" contains a list of changes to the APIs in the model (see column H). Afterwards all related documents (included in impact list) shall update their
generated artifacts.

General Requirements on Basic Software Modules
~~~~~~

Introduce the following requirements prior to SRS_BSW_00371:

SRS_BSW_xxxxx: Input parameters of scalar and enum types shall be passed as a value.
Type: valid
Description: All input parameters of scalar or enum type shall be passed as a value.
Rationale:

Use case: For example a function named <Mip>_SomeFunction with a return type of Std_ReturnType and a single parameter named SomeParameter of type uint8 is defined with the following signature:

Std_ReturnType <Mip>_SomeFunction(uint8 SomeParameter);
Dependencies: –
Supporting Material: —

SRS_BSW_yyyyy: Input parameters of structure type shall be passed as a reference to a constant structure

Type: valid

Description: All input parameters of structure type shall be passed as a reference constant structure

Rationale: Passing input parameters of structure type by value would result in additional run-time overhead due to efforts for copying the whole structure.

Use case: For example a function named <Mip>_SomeFunction with a return type of Std_ReturnType and a single parameter named SomeParameter of type SomeStructure (where SomeStructure is a struct) is defined with the following signature:

Std_ReturnType <Mip>_SomeFunction(P2CONST(SomeStructure, AUTOMATIC, <MIP>_APPL_DATA) SomeParameter);

Dependencies: –

Supporting Material: —

SRS_BSW_zzzzz: Input parameters of array type shall be passed as a reference to the constant array base type

Type: valid

Description: All input parameters of array type shall be passed as a reference to the constant array base type

Rationale: This effectively matches the behavior specified in the ISO-C:90 namely that a "declaration of a parameter as 'array of type' shall be adjusted to 'qualified pointer to
type'".

Use case: For example a function named <Mip>_SomeFunction with a return type of Std_ReturnType and a single parameter named SomeParameter of type array of uint8 is defined with the following signature:

Std_ReturnType   <Mip>_SomeFunction(P2CONST(uint8,   AUTOMATIC, <MIP>_APPL_DATA) SomeParameter);

Dependencies: –

Supporting Material: —

General Specification of Transformers
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

In SWS_Xfrm_00036 change

const <type>* dataElement

to

<paramtype> dataElement

and add the following to the where clause after the API table after the bullet
"type is data type of the data element
"

<paramtype> is derived from <type> according to the parameter passing rules
rules defined by the SRS BSW General (see SRS_BSW_xxxxx, SRS_BSW_yyyyy,
and SRS_BSW_zzzzz) and SWS BSW General (see SWS_BSW_00186 and
SWS_BSW_00187).

In SWS_Xfrm_00038 change

[<type> data_1,] ...
[<type> data_n]

to

[<paramtype> data_1,] ...
[<paramtype> data_n]

and add the following to the where clause after the API table after the bullet
"type is data type of the data element
"

<paramtype> is derived from <type> according to the parameter passing rules
rules defined by the SRS BSW General (see SRS_BSW_xxxxx, SRS_BSW_yyyyy,
and SRS_BSW_zzzzz) and SWS BSW General (see SWS_BSW_00186 and
SWS_BSW_00187).

The following paragraph shall then be removed:

For the arguments of ClientServerOperation which are handed over to the
transformer as data_1, ..., data_n the requirements to API parameters stated in
chapter API Parameters of [5, SWS RTE] are valid (especially [SWS_Rte_01017],
[SWS_Rte_01018] and [SWS_Rte_05107]).

In SWS_Xfrm_00040 change

[<originalData1>, ...
<originalDataN>]

to

[<paramtype> originalData1,] ...
[<paramtype> originalDataN]

and add the following to the where clause after the API table after the bullet
"type is data type of the data element
"

<paramtype> is derived from <type> according to the parameter passing rules
rules defined by the SRS BSW General (see SRS_BSW_xxxxx, SRS_BSW_yyyyy,
and SRS_BSW_zzzzz) and SWS BSW General (see SWS_BSW_00186 and
SWS_BSW_00187).

In SWS_Xfrm_00044 change

<type> *data_1, ...
<type> *data_n

to

[<paramtype> data_1,] ...
[<paramtype> data_n]

and add the following to the where clause after the API table after the bullet
"type is data type of the data element
"

<paramtype> is derived from <type> according to the parameter passing rules
rules defined by the SRS BSW General (see SRS_BSW_xxxxx, SRS_BSW_yyyyy,
and SRS_BSW_zzzzz) and SWS BSW General (see SWS_BSW_00186 and
SWS_BSW_00187).

The following paragraph shall then be removed:

For the arguments of ClientServerOperation which are handed over to the transformer as data_1, ..., data_n the requirements to API parameters stated in chapter API Parameters of [5, SWS RTE] are valid (especially [SWS_Rte_01017], [SWS_Rte_01018] and [SWS_Rte_05107]).

Speci?cation of SOME/IP Transformer
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

In SWS_SomeIpXf_00138 change

const <type>* dataElement

to

<paramtype> dataElement

and add the following to the where clause after the API table after the bullet "type is data type of the data element
"

<paramtype> is derived from <type> according to the parameter passing rules rules defined by the SRS BSW General (see SRS_BSW_xxxxx, SRS_BSW_yyyyy, and SRS_BSW_zzzzz) and SWS BSW General (see SWS_BSW_00186 and SWS_BSW_00187).

In SWS_SomeIpXf_00141 change

[<type> data_1,] ...
[<type> data_n]

to

[<paramtype> data_1,] ...
[<paramtype> data_n]

and add the following to the where clause after the API table after the bullet "type is data type of the data element
"

<paramtype> is derived from <type> according to the parameter passing rules rules defined by the SRS BSW General (see SRS_BSW_xxxxx, SRS_BSW_yyyyy,

and SRS_BSW_zzzzz) and SWS BSW General (see SWS_BSW_00186 and SWS_BSW_00187).

The following paragraph shall then be removed:

For the arguments of ClientServerOperation which are handed over to the transformer as data_1, ..., data_n the requirements to API parameters stated in chapter API Parameters of [5, SWS RTE] are valid (especially [SWS_Rte_01017], [SWS_Rte_01018] and [SWS_Rte_05107]).

In SWS_SomeIpXf_00145 change

<type> *data_1, ...
<type> *data_n

to

[<paramtype> data_1,] ...
[<paramtype> data_n]

and add the following to the where clause after the API table after the bullet "type is data type of the data element
"

<paramtype> is derived from <type> according to the parameter passing rules rules defined by the SRS BSW General (see SRS_BSW_xxxxx, SRS_BSW_yyyyy, and SRS_BSW_zzzzz) and SWS BSW General (see SWS_BSW_00186 and SWS_BSW_00187).

The following paragraph shall then be removed:

For the arguments of ClientServerOperation which are handed over to the transformer as data_1, ..., data_n the requirements to API parameters stated in chapter API Parameters of [5, SWS RTE] are valid (especially [SWS_Rte_01017], [SWS_Rte_01018] and [SWS_Rte_05107]).

Specification of COM Based Transformer
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

In SWS_ComXf_00007 change

const <type>* dataElement

to

<paramtype> dataElement

and add the following to the where clause after the API table after the bullet "type is data type of the data element
"

<paramtype> is derived from <type> according to the parameter passing rules rules defined by the SRS BSW General (see SRS_BSW_xxxxx, SRS_BSW_yyyyy, and SRS_BSW_zzzzz) and SWS BSW General (see SWS_BSW_00186 and SWS_BSW_00187).

Specification of Time Sync over Ethernet
~~~~~~

In SWS_EthTSyn_00040 make the parameter DataPtr of EthTSyn_RxIndication const.

Specification of SWS FlexRay Interface
~~~~~~

Change SWS_FrIf_05073 from
FrIf_NumOfStartupFramesPtr (IN)
to
FrIf_NumOfStartupFramesPtr (OUT)

Specification of ADC
~~~~~~
~[SWS_Adc_00419] Adc_SetupResultBuffer: change Adc_ValueGroupType* to const Adc_ValueGroupType*
~[SWS_Adc_00369] Adc_ReadGroup: move Adc_ValueGroupType * from Parameters (in) to Parameters (out)

There is no need to change parameter from IN to INOUT in Adc_SetupResultBuffer

Specification of Com
~~~~~~
Change type of parameter MetaData of Com_TriggerIPDUSendWithMetaData from uint8* to const uint8*

Specification of ComM
~~~~~~
no change required

Specification of Dem
~~~~~~
no change required

Specification of DLT
~~~~~~
no change required

Specification of DoIP
~~~~~~
From:
Std_ReturnType <User>_DoIPRoutingActivationConfirmation(boolean* Confirmed, uint8* ConfirmationReqData, uint8* ConfirmationResData)
Std_ReturnType <User>_DoIPRoutingActivationAuthentication(boolean* Authentified, uint8* AuthenticationReqData, uint8* AuthenticationResData)

To:
Std_ReturnType <User>_DoIPRoutingActivationConfirmation(boolean* Confirmed, const uint8* ConfirmationReqData, uint8* ConfirmationResData)
Std_ReturnType <User>_DoIPRoutingActivationAuthentication(boolean* Authentified, const uint8* AuthenticationReqData, uint8* AuthenticationResData)

Specification of E2ELibrary
~~~~~~
no change required

Specification of Eth
~~~~~~
no change required

Specification of EthIf
~~~~~~
no change required

Specification of EthSwitchDriver
~~~~~~
no change required

Specification of ICUDriver
~~~~~~
SWS_Icu_00201: Icu_StartTimestamp
Parameter (IN): Icu_ValueType* BufferPtr shall be changed to Parameters (out) type

Specification of LdCom
~~~~~~
[SWS_LDCOM_00027]: LdCom_CopyTxData
BufReq_ReturnType LdCom_CopyTxData(PduIdType id, const PduInfoType* info, RetryInfoType* retry, PduLengthType* availableDataPtr) shall be changed to
BufReq_ReturnType LdCom_CopyTxData(PduIdType id, const PduInfoType* info, const RetryInfoType* retry, PduLengthType* availableDataPtr)

[SWS_LDCOM_00036]: Rte_LdComCbkCopyTxData_<sn>
BufReq_ReturnType Rte_LdComCbkCopyTxData_<sn>(const PduInfoType* info, RetryInfoType* retry, PduLengthType* availableDataPtr) shall be changed to
BufReq_ReturnType Rte_LdComCbkCopyTxData_<sn>(const PduInfoType* info, const RetryInfoType* retry, PduLengthType* availableDataPtr)

Specification of Lin
~~~~~~
PduInfoPtr needs to be const in Std_ReturnType Lin_SendFrame(uint8 Channel, const Lin_PduType* PduInfoPtr)

Specification of PduR
~~~~~~
* PduR_<User:LoTp>CopyTxData
add const to "RetryInfoType* retry"

Specification of J1939Nm
~~~~~~
Change parameter 'name' of User_AddressClaimedIndication to type 'const uint8*'

Specification of SoAd
~~~~~~
=> everything already fixed with RfC 65633

Specification of SPIHandlerDriver
~~~~~~
==> nothing to change for SWS SPI

Specification of SynchronizedTimeBaseManager
~~~~~~
"StbM not affected. All issues listed in the WP-A attachment have been already
implemented by IT 69124 in context of RfC 65633"

Specification of TcpIp
~~~~~~
~[SWS_TCPIP_00040] TcpIp_DhcpReadOption: change DataPtr from (IN) to
(OUT)
~[SWS_TCPIP_00189] TcpIp_DhcpV6ReadOption: change DataPtr from (IN) to
(OUT)
=> everything else already fixed with RfC 65633

Specification of TimeSyncOverFlexRay
~~~~~~
"Change SWS_FrTSyn_00064: parameter versioninfo of type Std_VersionInfoType*
is marked wrongly as IN. Change to OUT"

Specification of EFX

~~~~~~~~~~~~~~~~~~~~~

~ [SWS_Efx_00355] Efx_Debounce_u8_u8: Include constant for pointer Input-parameter as like below.
uint8 Efx_Debounce_u8_u8(boolean X, Efx_DebounceState_Type * State, const Efx_DebounceParam_Type * Param, sint32 dT)

~ [SWS_Efx_00376] Efx_MedianSort: The parameter <InType>* Array should be InOut instead of In parameter as like below.
Parameters (in): N Size of an array
Parameters (inout): Array Pointer to an array

~ [SWS_Efx_00309] Efx_RampCheckActivity: Include constant for pointer Input-parameter as like below.
boolean Efx_RampCheckActivity(const Efx_StateRamp_Type* State_cpst)

~ [SWS_Efx_00307] Efx_RampGetSwitchPos: Include constant for pointer Input-parameter as like below.
boolean Efx_RampGetSwitchPos(const Efx_StateRamp_Type* State_cpst)

~ [SWS_Efx_00193] Efx_Array_Average: Include constant for pointer Input-parameter as like below.
<OutType> Efx_Array_Average_<InTypeMn>_<OutTypeMn>(const <InType>* Array, uint16 Count)

Specification of MFL
~~~~~~~~~~~~~~~~~~~~~

~ [SWS_Mfl_00192] Mfl_Debounce_u8_u8: Include constant for pointer Input-parameter as like below.
boolean Mfl_Debounce_u8_u8(boolean X, Mfl_DebounceState_Type* State, const Mfl_DebounceParam_Type* Param, float32 dT)

~ [SWS_Mfl_00266] Mfl_DebounceInit: The parameter Mfl_DebounceState_Type* State should be Out instead of In parameter as like below.
Parameters (in): X Initial value for the input state
Parameters (out): State Pointer to structure for debouncing state variables

~ [SWS_Mfl_00246] Mfl_HystDeltaRight_f32_u8: Include constant for pointer Input-parameter as like below.
boolean Mfl_HystDeltaRight_f32_u8(float32 X, float32 Delta, float32 Rsp, const uint8* State)

~ [SWS_Mfl_00285] Mfl_MedianSort_f32_f32: The parameter Array should be

InOut instead of In parameter as like below.
Parameters (in): N Size of an array
Parameters (inout): Array Pointer to an array

~ [SWS_Mfl_00037] Mfl_PT1SetState: The parameter State_cpst should be Out instead of In parameter as like below.
Parameters (in): X1_f32 Initial value for input state
Y1_f32 Initial value for output state
Parameters (out): State_cpst Pointer to internal state structure

~ [SWS_Mfl_00225] Mfl_RampCheckActivity: Include constant for pointer Input-parameter as like below.
boolean Mfl_RampCheckActivity(const Mfl_StateRamp_Type* State_cpst)

~ [SWS_Mfl_00223] Mfl_RampGetSwitchPos: Include constant for pointer Input-parameter as like below.
boolean Mfl_RampGetSwitchPos(const Mfl_StateRamp_Type* State_cpst)
–Last change on issue 68035 comment 135–

BW-C-Level:

Application	Specification	Bus
1	4	1