

Document Title	SWS_SOMEIPTransformer: 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_SOMEIPTransformer	3
1.1	Specification Item SWS_SomelpXf_00017	3
1.2	Specification Item SWS_SomelpXf_00107	4
1.3	Specification Item SWS_SomelpXf_00138	6
1.4	Specification Item SWS_SomelpXf_00141	19
1.5	Specification Item SWS_SomelpXf_00145	33
1.6	Specification Item SWS_SomelpXf_00206	46
1.7	Specification Item SWS_SomelpXf_00212	48
1.8	Specification Item SWS_SomelpXf_00228	49
1.9	Specification Item SWS_SomelpXf_00229	61
1.10	Specification Item SWS_SomelpXf_00232	73
1.11	Specification Item SWS_SomelpXf_00244	85
1.12	Specification Item SWS_SomelpXf_00245	86
1.13	Specification Item SWS_SomelpXf_00246	88
1.14	Specification Item SWS_SomelpXf_00247	90
1.15	Specification Item SWS_SomelpXf_00249	92

1 SWS_SOMEIPTransformer

1.1 Specification Item SWS_SomelpXf_00017

Trace References:

SRS_Xfrm_00101

Content:

If less data than expected are handed over to the SOME/IP transformer during deserialization of data, the following shall happen:

- if for the corresponding ISignal an initial value is specified (in serialized form) use **that** the value to fill the missing elements **at the end of the serialized stream**.
- if no initial value is available abort deserialization with `E_SER_MALFORMED_MESSAGE`.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #76270: Improve Support For ISignal Init Value For `E_SER_MALFORMED_MESSAGE`

Problem description:

SWS Requirement [SWS_SomelpXf_00017]: If less data than expected are handed over to the SOME/IP transformer during deserialization of data, the following shall happen:
if for the corresponding ISignal an initial value is specified (in serialized form) use that value to fill the missing elements.
if no initial value is available abort deserialization with `E_SER_MALFORMED_MESSAGE`.
(SRS_Xfrm_00101)

Part of the requirement i.e. if for the corresponding ISignal an initial value is specified (in serialized form) use that value to fill the missing elements is ambiguous.

Information about which elements are missing is not available at the receiver and hence filling of the missing elements with initial value could lead to incorrect reception.

e.g.
Struct Test1

uint8 ele1

uint16 ele2
uint64 ele3

Then Serialized Buffer at sender will be :
buffer = header,ele1,ele2,ele3;

At deserialization time at receiver if ele2 is missing in buffer, it will not be known and initial value could be filled wrongly at the end.

ele1 = received first byte,

ele2 = next received 2 bytes

ele3 = next received 6 bytes + initial value of ISignal;

This deserialized data will be incorrect.

Agreed solution:

Change SWS_SomelpXf_00017 from:

"[...] if for the corresponding ISignal an initial value is specified (in serialized form) use that value to fill the missing elements. [...]"

to

"[...] if for the corresponding ISignal an initial value is specified (in serialized form) use the value to fill the missing elements at the end of the serialized stream. [...]"

Add hint below SWS_SomelpXf_00017:

"Missing data can only be recognized by comparing the length of received serialized data with the expected length of the data.

SWS_SomelpXf_00017 enables extensions of data by adding elements to the end and achieve backward compatibility of an ECU with older boardnet layouts that are missing those data."

—Last change on issue 76270 comment 3—

BW-C-Level:

Application	Specification	Bus
1	1	1

1.2 Specification Item SWS_SomelpXf_00107

Trace References:

SRS_Xfrm_00102

Content:

The SOME/IP transformer on the server-ECU builds its header for the server response based on the header of the client's request and does in addition:

- Construct the payload
- Set the Message Type to |RESPONSE| (i.e. |0x80|)
- If the ClientServerOperation has at least one ClientServerOperation.possibleError defined, place the return value of the executed ClientServerOperation into the Return Code field (see chapter [REF sec_3a_ReturnCode]) and add |0x1F| to adapt the number ranges . Use in case the original return value was different from |E_OK| in the SOME/IP header if the ClientServerOperation has no ClientServerOperation.possibleError defined|0x00|.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #76926: Contradiction between SWS_SomelpXf_00107 and constr_1108

Problem description:

The issue is in relation to https://www.autosar.org/bugzilla/show_bug.cgi?id=72952

According to constr_1108 within the Software Component Template AUTOSAR CP Release 4.3.0 it is possible to define a possibleError which represents E_OK with value 0.

According to SWS_SomelpXf_00107 :

' If the ClientServerOperation has at least one possibleError defined, place the return value of the executed ClientServerOperation into the Return Code field (see chapter 7.2.3.7) and add 0x1F to adapt the number ranges. Use E_OK in the SOME/IP header if the ClientServerOperation has no possibleError defined.'

The Return Code field values are specified with SWS_SomelpXf_00115 and include 0x00 with E_OK but for application errors the value range of 0x20-0x5e is specified.

If the return value is E_OK and 0x1F will be added the value 0x1F is not in the value range of the application errors 0x20-0x5e within the Return Code field values.

How should the Return Code field be adapted in case of return value E_OK?

IMHO, the right behaviour is that the Return Code field value shall only be adapted by adding 0x1F if the ClientServerOperation has at least one possibleError defined and the return value of the executed ClientServerOperation is not E_OK (0).

Agreed solution:

Change last part of SWS_SomelpXf_00107 to:

If the ClientServerOperation has at least one possibleError defined, place the return value of the executed ClientServerOperation into the Return Code field (see chapter 7.2.3.7) and add 0x1F to adapt the number ranges in case the original return value was different from 0x00.

–Last change on issue 76926 comment 3–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.3 Specification Item SWS_SomelpXf_00138**Trace References:**

none

Content:

Service name:	SomelpXf_<transformerId> SomelpXf_transformerId1	
Syntax:	uint8 SomelpXf_<transformerId>(uint8* buffer, uint32* bufferLength, const <typeparamtype> * dataElement)	
Service ID[hex]:	0x03	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	dataElement SomelpXf_transformerId1.dataElement	Data element which shall be transformed
Parameters (inout):	None	
Parameters (out):	buffer SomelpXf_transformerId1.buffer	Buffer allocated by the RTE, where the transformed data has to be stored by the transformer
	bufferLength SomelpXf_transformerId1.bufferLength	Used length of the buffer
Return value:	uint8	0x00 (E_OK): Serialization successful 0x81 (E_SER_GENERIC_ERROR): A generic error occurred
Description:	This function transforms a Sender/Receiver communication using the serialization of SOME/IP. It takes the data element as input and outputs a uint8 array containing the serialized data. The length of the serialized data shall be calculated by the transformer during runtime and returned in the OUT-parameter bufferLength. It may be smaller than the maximum buffer size used by the RTE for buffer allocation.	

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]).

Specification 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 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 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_Frlf_05073 from
Frlf_NumOfStartupFramesPtr (IN)
to
Frlf_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* Authenticated, uint8* AuthenticationReqData, uint8* AuthenticationResData)

To:

Std_ReturnType <User>_DoIPRoutingActivationConfirmation(boolean* Confirmed, const uint8* ConfirmationReqData, uint8* ConfirmationResData)

Std_ReturnType <User>_DoIPRoutingActivationAuthentication(boolean* Authenticated, 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_LdComCbkJCopyTxData_<sn>

BufReq_ReturnType Rte_LdComCbkJCopyTxData_<sn>(const PduInfoType* info, RetryInfoType* retry, PduLengthType* availableDataPtr) shall be changed to

BufReq_ReturnType Rte_LdComCbkJCopyTxData_<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 Tcplp

~~~~~

~[SWS\_TCPIP\_00040] Tcplp\_DhcpReadOption: change DataPtr from (IN) to (OUT)

~[SWS\_TCPIP\_00189] Tcplp\_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

- RfC #74087: Change "an uint" to "a uint"

Problem description:

Remainder from # 73404:

The affected documents contain text generated artefacts which contain the text "an uint".

Correct is "a uint".

The changes of the artefacts need changes in Metamodel and BSW UML Model.

Agreed solution:

Change "an uint" to "a uint" in metamodel artifacts.

–Last change on issue 74087 comment 2–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.4 Specification Item SWS_SomelpXf_00141

Trace References:

none

Content:

Service name:	SomelpXf_<transformerId> SomelpXf_transformerId2	
Syntax:	<pre>uint8 SomelpXf_<transformerId>(const Rte-Cs_TransactionHandleType* TransactionHandle, uint8* buffer, uint32* bufferLength, [Std_ReturnType returnValue,] <typeparamtype> data_1, ... <typeparamtype> data_n)</pre>	
Service ID[hex]:	0x03	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	TransactionHandle SomelpXf_transformerId2.TransactionHandle	Transaction handle according to [SWS_Rte_08732] (clientId and sequenceCounter) needed to differentiate between multiple requests.
	returnValue SomelpXf_transformerId2.returnValue	Return value of the server runnable which needs to be serialized on server from server side for transmission to the calling client. This argument is only available for serializers of the response of a Client/Server communication and if - the ClientServerOperation has at least one PossibleError defined . or - autonomous error reaction is activated
	data_1 SomelpXf_transformerId2.data_1	Client/Server operation argument which shall be transformed (in the same order as in the corresponding interface)
... SomelpXf_transformerId2....	...	
data_n SomelpXf_transformerId2.data_n	Client/Server operation argument which shall be transformed (in the same order as in the corresponding interface)	
Parameters (inout):	None	
Parameters (out):	buffer SomelpXf_transformerId2.buffer	Buffer allocated by the RTE, where the transformed data has to be stored by the transformer
	bufferLength SomelpXf_transformerId2.bufferLength	Used length of the buffer
Return value:	uint8	0x00 (E_OK): Serialization successful 0x81 (E_SER_GENERIC_ERROR): A generic error occurred

Description:	This function transforms a Client/Server communication using the serialization of SOME/IP. It takes the operation arguments and optionally the return value as input and outputs a uint8 array containing the serialized data. The length of the serialized data shall be calculated by the transformer during runtime and returned in the OUT-parameter bufferLength. It may be smaller than the maximum buffer size used by the RTE for buffer allocation.
--------------	--

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 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 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 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_Frlf_05073 from
Frlf_NumOfStartupFramesPtr (IN)
to
Frlf_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* Authenticated, 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_LdComCbkJCopyTxData_<sn>
BufReq_ReturnType Rte_LdComCbkJCopyTxData_<sn>(const PduInfoType* info,
RetryInfoType* retry, PduLengthType* availableDataPtr) shall be changed to
BufReq_ReturnType Rte_LdComCbkJCopyTxData_<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 Tcplp

~~~~~

~[SWS\_TCPIP\_00040] Tcplp\_DhcpReadOption: change DataPtr from (IN) to  
(OUT)

~[SWS\_TCPIP\_00189] Tcplp\_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

- RfC #74087: Change "an uint" to "a uint"

Problem description:

Remainder from # 73404:

The affected documents contain text generated artefacts which contain the text "an uint".

Correct is "a uint".

The changes of the artefacts need changes in Metamodel and BSW UML Model.

Agreed solution:

Change "an uint" to "a uint" in metamodel artifacts.

–Last change on issue 74087 comment 2–

BW-C-Level:

Application	Specification	Bus
1	1	1

- RfC #76702: Wrong description of parameter returnValue for client/server response

Problem description:

The input parameter returnValue of the SomelpXf_<transformerId> specified by SWS_SomelpXf_00141 is given by:

Return value of the server runnable which needs to be serialized on server side for transmission to the calling client. This argument is only available for serializers of the response of a Client/Server communication and if the ClientServerOperation has at least one PossibleError defined.

This description became false with the introduction of autonomous error response by RfC 68623, see

https://www.autosar.org/bugzilla/show_bug.cgi?id=68623 .

For autonomous error response, no runnable is necessary and also no PossibleError needs to be defined.

Agreed solution:

Replace former description of parameter returnValue (SWS_SomelpXf_00141 and SWS_SomelpXf_00145) by:

Return value from server side for transmission to the calling client. This argument is only available for serializers of the response of a Client/Server communication if

- the ClientServerOperation has at least one PossibleError defined or
- autonomous error reaction is activated

–Last change on issue 76702 comment 1–

BW-C-Level:

Application	Specification	Bus
4	4	1

1.5 Specification Item SWS_SomelpXf_00145

Trace References:

none

Content:

Service name:	SomelpXf_Inv_<transformerId>SomelpXf_Inv_transformerId2	
Syntax:	<pre>uint8 SomelpXf_Inv_<transformerId>(Rte-Cs_TransactionHandleType* TransactionHandle, const uint8* buffer, uint32 bufferLength, [Std_ReturnType* returnValue,] [<typeparamtype> * data_1,] ... [<typeparamtype> * data_n])</pre>	
Service ID[hex]:	0x04	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	bufferSomelpXf_Inv_transformerId2.buffer	Buffer allocated by the RTE, where the still serialized data are stored by the Rte
	bufferLengthSomelpXf_Inv_transformerId2.bufferLength	Used length of the buffer
Parameters (inout):	None	
Parameters (out):	TransactionHandleSomelpXf_Inv_transformerId2.TransactionHandle	Transaction handle according to [SWS_Rte_08732] (clientId and sequenceCounter) needed to differentiate between multiple requests.
	returnValueSomelpXf_Inv_transformerId2.returnValue	Return value of the server runnable which needs to be serialized on server from server side for transmission to the calling client. This argument is only available for deserializers serializers of the response of a Client/Server communication and if - the Client ServerOperation has at least one PossibleError defined . or - autonomous error reaction is activated
	data_1SomelpXf_Inv_transformerId2.data_1	Client/Server operation argument which shall be transformed (in the same order as in the corresponding interface)
...SomelpXf_Inv_transformerId2....	...	
data_nSomelpXf_Inv_transformerId2.data_n	Client/Server operation argument which shall be transformed (in the same order as in the corresponding interface)	

Return value:	uint8	0x00 (E_OK): Deserialization successful 0x01 (E_NO_DATA): No data available which can be deserialized 0x81 (E_SER_GENERIC_ERROR): A generic error occurred 0x87 (E_SER_WRONG_PROTOCOL_VERSION): The version of the receiving transformer didn't match the sending transformer. 0x88 (E_SER_WRONG_INTERFACE_VERSION): Interface version of serialized data is not supported. 0x89 (E_SER_MALFORMED_MESSAGE): The received message is malformed. The transformer is not able to produce an output. 0x8a (E_SER_WRONG_MESSAGE_TYPE): The received message type was not expected.
Description:	This function deserializes a Client/Server communication using the deserialization of SOME/IP. It takes the uint8 array containing the serialized data as input and outputs the return value of the server runnable and the operation arguments which have to be passed from the server to the client.	

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\_SomelpXf\_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_Frlf_05073 from
Frlf_NumOfStartupFramesPtr (IN)
to
Frlf_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 Param-  
eters (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_LdComCbkJCopyTxData_<sn>

BufReq_ReturnType Rte_LdComCbkJCopyTxData_<sn>(const PduInfoType* info,
RetryInfoType* retry, PduLengthType* availableDataPtr) shall be changed to
BufReq_ReturnType Rte_LdComCbkJCopyTxData_<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

- RfC #76702: Wrong description of parameter returnValue for client/server response

Problem description:

The input parameter returnValue of the SomeIpXf_<transformerId> specified by SWS_SomeIpXf_00141 is given by:

Return value of the server runnable which needs to be serialized on server side for transmission to the calling client. This argument is only available for serializers of the response of a Client/Server communication and if the ClientServerOperation has at least one PossibleError defined.

This description became false with the introduction of autonomous error response by RfC 68623, see

https://www.autosar.org/bugzilla/show_bug.cgi?id=68623 .

For autonomous error response, no runnable is necessary and also no PossibleError needs to be defined.

Agreed solution:

Replace former description of parameter returnValue (SWS_SomeIpXf_00141 and SWS_SomeIpXf_00145) by:

Return value from server side for transmission to the calling client. This argument is only available for serializers of the response of a Client/Server communication if

- the ClientServerOperation has at least one PossibleError defined or
- autonomous error reaction is activated

–Last change on issue 76702 comment 1–

BW-C-Level:

Application	Specification	Bus
4	4	1

1.6 Specification Item SWS_SomeIpXf_00206

Trace References:

SRS_Xfrm_00002

Content:

Service name:	SomeIpXf_<transformerId>
---------------	--------------------------

Syntax:	uint8 SomeIpXf_<transformerId>(uint8* buffer, uint32* bufferLength)	
Service ID[hex]:	0x03	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	buffer	Buffer allocated by the RTE, where the transformed data has to be stored by the transformer
	bufferLength	Used length of the buffer
Return value:	uint8	0x00 (E_OK): Serialization successful 0x81 (E_SER_GENERIC_ERROR): A generic error occurred
Description:	This function transforms an external trigger event using the serialization of SOME/IP. It takes trigger as input and outputs an uint8 array. The length of the transformed data shall be calculated by the transformer during runtime and returned in the OUT parameter bufferLength. It may be smaller than the maximum buffer size used by the RTE for buffer allocation.	

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #74087: Change "an uint" to "a uint"

Problem description:

Remainder from # 73404:

The affected documents contain text generated artefacts which contain the text "an uint".

Correct is "a uint".

The changes of the artefacts need changes in Metamodel and BSW UML Model.

Agreed solution:

Change "an uint" to "a uint" in metamodel artifacts.

–Last change on issue 74087 comment 2–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.7 Specification Item SWS_SomelpXf_00212

Trace References:

SRS_Xfrm_00008

Content:

One Session Handling ID counter (32 16 Bit) has to be maintained per transformer function for Sender/Receiver communication (see SWS_SomelpXf_00138) if SOMEIPTransformationDescription.sessionHandlingSR is set to SOMEIPTransformerSessionHandlingEnum.sessionHandlingActive.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77008: Size of " Session Handling ID counter" is 32 Bit according to SWS_SomelpXf_00212

Problem description:

Name: Robert Sakretz

Phone:

Role: Implementor

Description/Motivation:

According to SWS_SomelpXf_00024 the Session Id is 16 bit and part of the Request Id.

But this statement is confusing:

[SWS_SomelpXf_00212] One Session Handling ID counter (32 Bit) has to be maintained per transformer function for Sender/Receiver communication (see [SWS_SomelpXf_00138]) if sessionHandlingSR is set to sessionHandlingActive.

and according to [SWS_SomelpXf_00215] the value of the Session Handling ID counter shall wrap around at 0xFFFF (16 bit).

So how shall the Session Id be used?

Agreed solution:

Change [SWS_SomelpXf_00212] to:

"[...] counter (16 Bit) has to [...]"

–Last change on issue 77008 comment 2–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.8 Specification Item SWS_SomelpXf_00228

Trace References:

none

Content:

In function |SomelpXf_<transformerId>| defined in SWS_SomelpXf_00138

- |paramtype| is derived from |type| according to the parameter passing rules defined by the SRS_2d_SRSBSWGeneral (see SRS_BSW_00484, SRS_BSW_00485, and SRS_BSW_00486) and SWS_2d_BSWGeneral (see SWS_BSW_00186 and SWS_BSW_00187).
- |type| shall be the data type of the data element after all data conversion activities of the RTE
- |transformerId| shall be the name pattern for the transformer specified in SWS_Xfrm_00062 (ASWS_2d_TransformerGeneral)

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\_SomelpXf\_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 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 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_Frlf_05073 from
Frlf_NumOfStartupFramesPtr (IN)
to
Frlf_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* 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_LdComCbkJCopyTxData_<sn>
BufReq_ReturnType Rte_LdComCbkJCopyTxData_<sn>(const PduInfoType* info, RetryInfoType* retry, PduLengthType* availableDataPtr) shall be changed to
BufReq_ReturnType Rte_LdComCbkJCopyTxData_<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 Tcplp

~~~~~

~[SWS\_TCPIP\_00040] Tcplp\_DhcpReadOption: change DataPtr from (IN) to (OUT)

~[SWS\_TCPIP\_00189] Tcplp\_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

1.9 Specification Item SWS_SomelpXf_00229

Trace References:

none

Content:

In function |SomelpXf_<transformerId>| defined in SWS_SomelpXf_00141

- |paramtype| is derived from |type| according to the parameter passing rules defined by the SRS_2d_SRSBSWGeneral (see SRS_BSW_00484, SRS_BSW_00485, and SRS_BSW_00486) and SWS_2d_BSWGeneral (see SWS_BSW_00186 and SWS_BSW_00187).
- |type| shall be the data type of the data element after all data conversion activities of the RTE
- |transformerId| shall be the name pattern for the transformer specified in SWS_Xfrm_00062 (ASWS_2d_TransformerGeneral).

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\_SomelpXf\_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_Frlf_05073 from
Frlf_NumOfStartupFramesPtr (IN)
to
Frlf_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 Tcplp

~~~~~

~[SWS\_TCPIP\_00040] Tcplp\_DhcpReadOption: change DataPtr from (IN) to (OUT)

~[SWS\_TCPIP\_00189] Tcplp\_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

1.10 Specification Item SWS_SomelpXf_00232

Trace References:

none

Content:

In function |SomelpXf_Inv_<transformerId>| defined in SWS_SomelpXf_00145

- |paramtype| is derived from |type| according to the parameter passing rules defined by the SRS_2d_SRSBSWGeneral (see SRS_BSW_00484, SRS_BSW_00485, and SRS_BSW_00486) and SWS_2d_BSWGeneral (see SWS_BSW_00186 and SWS_BSW_00187).
- |type| shall be the data type of the data element before all data conversion activities of the RTE
- |transformerId| shall be the name pattern for the transformer specified in SWS_Xfrm_00062 (ASWS_2d_TransformerGeneral).

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 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_Frlf_05073 from
Frlf_NumOfStartupFramesPtr (IN)
to
Frlf_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* 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 Tcplp

~~~~~

~[SWS\_TCPIP\_00040] Tcplp\_DhcpReadOption: change DataPtr from (IN) to (OUT)

~[SWS\_TCPIP\_00189] Tcplp\_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

1.11 Specification Item SWS_SomelpXf_00244

Trace References:

SRS_Xfrm_00101

Content:

Serialization of fixed length strings shall consist of the following steps:

1. Appending BOM at the beginning of the output buffer, if BOM is not already available in the first three (UTF-8) or two (UTF-16) bytes of the to be serialized array containing the string. If the BOM is already present, simply copy the BOM into the output buffer.
2. Copying the string data (**until a string terminator is reached but at most** the number of bytes according to the string's fixed length) from the array into the output buffer, optionally performing a conversion between UTF-16LE and UTF-16BE between ECU and network byte order if BaseTypeDirectDefinition.BaseTypeDirectDefinition.byteOrder and SOMEIPTransformationDescription.SOMEIPTransformationDescription.byteOrder have different values
3. Termination of the string with |0x00| (UTF-8) or |0x0000| (UTF-16) if not yet terminated - Note that this basically means that |0x00| (UTF-8) is written into the last byte or |0x0000| (UTF-16) into the last two bytes of the fixed length string
4. **Padding of unused memory in strings with |0x00| in case the string has already been terminated.**

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77384: Facilitate exchange of strings containing the end-of-string character '\{0}'

Problem description:

In C++ std::strings may contain end-of-string characters (i.e., '\{0}') since the length of the string is stored explicitly. As far as copying of std::strings is concerned,

any contained end-of-string character is handled in the same way as any other character. - Thus for the AUTOSAR AP it makes sense to mimic this behavior in case std::strings are transferred over the network.

To achieve a consistent behavior between receiving AUTOSAR CP applications and receiving AUTOSAR AP application, the handling of contained end-of-string characters in AUTOSAR CP shall be aligned to that of AUTOSAR AP.

Agreed solution:

SWS_SomelpXf_00244.2: Remove the until a string terminator is reached part.

SWS_SomelpXf_00244.4: Remove this sub-item

SWS_SomelpXf_00246.6: Remove the until a string terminator is reached part.

SWS_SomelpXf_00245.1/2: Switch the two sub-items to make it clear that the length field goes first (no matter if the BOM needs to be added or not)

SWS_SomelpXf_00245.3: Remove the until a string terminator is reached part.

SWS_SomelpXf_00245.5: Remove this sub-item

SWS_SomelpXf_00247.6: Remove the until a string terminator is reached part.

–Last change on issue 77384 comment 1–

BW-C-Level:

Application	Specification	Bus
1	4	4

1.12 Specification Item SWS_SomelpXf_00245

Trace References:

SRS_Xfrm_00101

Content:

Serialization of dynamic length strings shall consist of the followign steps:

1. Appending BOM at the beginning, if BOM is not already available in the first 3 (UTF-8) or 2 (UTF-16) bytes of the to be serialized array containing the string. If the BOM is already present, simply copy the BOM into the output buffer

2. ~~Sdd the~~ **Add the** Length Field - The value of the length field shall be computed by multiplying the number of elements given by the size indicator with the size in bytes of each element (i.e., 1 for UTF-8 and 2 for UTF-16) increased by the size in bytes needed the BOM. **The data type of the length field shall be determined from the size indicator ImplementationDataTypeElement that points to a uint8, uint16 or uint32.**
3. **Appending BOM at the beginning, if BOM is not already available in the first 3 (UTF-8) or 2 (UTF-16) bytes of the to be serialized array containing the string. If the BOM is already present, simply copy the BOM into the output buffer**
4. Copying the string data (~~until a string terminator is reached but at most the~~ **copy the** the number of bytes according to the string's size indicator and the size of bytes of each element) from the array into the output buffer, optionally performing a conversion between UTF-16LE and UTF-16BE between ECU and network byte order Base TypeDirectDefinition.BaseTypeDirectDefinition.byteOrder and SOMEIPTransformationDescription.SOMEIPTransformationDescription.byteOrder have different values
5. Termination of the string with |0x00| (UTF-8) or |0x0000| (UTF-16) if not terminated yet - Note that this basically means that |0x00| (UTF-8) is written into the last byte or |0x0000| (UTF-16) into the last two bytes (according to the length indicator) of the variable length string
6. **Padding of unused memory in strings with |0x00| in case the string has already been terminated**

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77384: Facilitate exchange of strings containing the end-of-string character '\0'

Problem description:

In C++ std::strings may contain end-of-string characters (i.e., '\0') since the length of the string is stored explicitly. As far as copying of std::strings is concerned, any contained end-of-string character is handled in the same way as any other character. - Thus for the AUTOSAR AP it makes sense to mimic this behavior in case std::strings are transferred over the network.

To achieve a consistent behavior between receiving AUTOSAR CP applications and receiving AUTOSAR AP application, the handling of contained end-of-string characters in AUTOSAR CP shall be aligned to that of AUTOSAR AP.

Agreed solution:

SWS_SomeIpXf_00244.2: Remove the until a string terminator is reached part.

SWS_SomeIpXf_00244.4: Remove this sub-item

SWS_SomelpXf_00246.6: Remove the until a string terminator is reached part.

SWS_SomelpXf_00245.1/2: Switch the two sub-items to make it clear that the length field goes first (no matter if the BOM needs to be added or not)

SWS_SomelpXf_00245.3: Remove the until a string terminator is reached part.

SWS_SomelpXf_00245.5: Remove this sub-item

SWS_SomelpXf_00247.6: Remove the until a string terminator is reached part.

–Last change on issue 77384 comment 1–

BW-C-Level:

Application	Specification	Bus
1	4	4

- RfC #77439: [SWS_SomelpXf_00245] shall be more precise in regard to the length field

Problem description:

[SWS_SomelpXf_00245] defines the serialization of dynamic length strings.

In this specification item a point is missing that explains that the data type of the SOME/IP length field is determined from the size indicator Implementation-Data-DataElement that points to a uint8, uint16 or uint32.

Agreed solution:

Change point 2 of [SWS_SomelpXf_00245]

- Change "Sdd" to "Add" (typo!)

- Add to the end of point 2:

"The data type of the length field shall be determined from the size indicator Implementation-Data-DataElement that points to a uint8, uint16 or uint32."

–Last change on issue 77439 comment 2–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.13 Specification Item SWS_SomelpXf_00246

Trace References:

SRS_Xfrm_00101

Content:

Deserialization of fixed length strings shall consist of the following steps:

1. Check whether the string starts with a BOM. If not, a |MALFORMED_MESSAGE| error shall be issued
2. Check whether BOM has the same value as SOMEIPTransformationDescription.SOMEIPTransformationDescription.byteOrder. If not, a |MALFORMED_MESSAGE| error shall be issued
3. Remove the BOM
4. Silently discard the last byte of the string in case of an UTF-16 string with odd length
5. Check whether the string terminates with |0x00| (UTF-8) or |0x0000| (UTF-16). If not, a |MALFORMED_MESSAGE| error shall be issued
6. Copy the string data (until a string terminator is reached but at most the number of bytes according to the string's fixed length) from the input buffer into the array, optionally performing a conversion between UTF-16LE and UTF-16BE between network and ECU byte order if BaseTypeDirectDefinition.BaseTypeDirectDefinition.byteOrder and SOMEIPTransformationDescription.SOMEIPTransformationDescription.byteOrder have different values.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77384: Facilitate exchange of strings containing the end-of-string character '\{0}'

Problem description:

In C++ std::strings may contain end-of-string characters (i.e., '\{0}') since the length of the string is stored explicitly. As far as copying of std::strings is concerned, any contained end-of-string character is handled in the same way as any other character. - Thus for the AUTOSAR AP it makes sense to mimic this behavior in case std::strings are transferred over the network.

To achieve a consistent behavior between receiving AUTOSAR CP applications and receiving AUTOSAR AP application, the handling of contained end-of-string characters in AUTOSAR CP shall be aligned to that of AUTOSAR AP.

Agreed solution:

SWS_SomeIPXf_00244.2: Remove the until a string terminator is reached part.
SWS_SomeIPXf_00244.4: Remove this sub-item

SWS_SomelpXf_00246.6: Remove the until a string terminator is reached part.

SWS_SomelpXf_00245.1/2: Switch the two sub-items to make it clear that the length field goes first (no matter if the BOM needs to be added or not)

SWS_SomelpXf_00245.3: Remove the until a string terminator is reached part.

SWS_SomelpXf_00245.5: Remove this sub-item

SWS_SomelpXf_00247.6: Remove the until a string terminator is reached part.

–Last change on issue 77384 comment 1–

BW-C-Level:

Application	Specification	Bus
1	4	4

1.14 Specification Item SWS_SomelpXf_00247

Trace References:

SRS_Xfrm_00101

Content:

Deserialization of dynamic length strings shall consist of the following steps:

1. Check whether the string starts with a BOM. If not, a |MALFORMED_MESSAGE| error shall be issued
2. Check whether BOM has the same value as SOMEIPTransformationDescription.SOMEIPTransformationDescription.byteOrder. If not, a |MALFORMED_MESSAGE| error shall be issued
3. Remove the BOM and reduce the value of the length field accordingly
4. Silently discard the last byte of the string in case of an UTF-16 string with odd length (according to the reduced value of the length field)
5. Check whether the string terminates with |0x00| (UTF-8) or |0x0000| (UTF-16). If not, a |MALFORMED_MESSAGE| error shall be issued
6. Copy the string data (until a string terminator is reached but at most copy the number of bytes according to the string's reduced value of the length field) from the input buffer into the array, optionally performing a conversion between UTF-

16LE and UTF-16BE between network and ECU byte order if BaseTypeDirectDefinition.byteOrder and SOMEIPTransformationDescription.byteOrder have different values, optionally performing a conversion between UTF-16LE and UTF-16BE between ECU and bus if BaseTypeDirectDefinition.BaseTypeDirectDefinition.byteOrder and SOMEIPTransformationDescription.SOMEIPTransformationDescription.byteOrder have different values

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77384: Facilitate exchange of strings containing the end-of-string character '\0'

Problem description:

In C++ std::strings may contain end-of-string characters (i.e., '\0') since the length of the string is stored explicitly. As far as copying of std::strings is concerned, any contained end-of-string character is handled in the same way as any other character. - Thus for the AUTOSAR AP it makes sense to mimic this behavior in case std::strings are transferred over the network.

To achieve a consistent behavior between receiving AUTOSAR CP applications and receiving AUTOSAR AP application, the handling of contained end-of-string characters in AUTOSAR CP shall be aligned to that of AUTOSAR AP.

Agreed solution:

SWS_SomelpXf_00244.2: Remove the until a string terminator is reached part.

SWS_SomelpXf_00244.4: Remove this sub-item

SWS_SomelpXf_00246.6: Remove the until a string terminator is reached part.

SWS_SomelpXf_00245.1/2: Switch the two sub-items to make it clear that the length field goes first (no matter if the BOM needs to be added or not)

SWS_SomelpXf_00245.3: Remove the until a string terminator is reached part.

SWS_SomelpXf_00245.5: Remove this sub-item

SWS_SomelpXf_00247.6: Remove the until a string terminator is reached part.

–Last change on issue 77384 comment 1–

BW-C-Level:

Application	Specification	Bus
1	4	4

1.15 Specification Item SWS_SomeIpXf_00249

Trace References:

SRS_Xfrm_00101

Content:

A union shall be detected if an ImplementationDataType with the following pattern (named wrapped union data type) is used: ImplementationDataType with category STRUCTURE that contains exactly two ImplementationDataTypeElements:

- memberSelector: ImplementationDataTypeElement which represents the type field that boils down to **an** uint8, uint16 or uint32 ImplementationDataType
- payload: ImplementationDataTypeElement of category UNION which represents the actual union

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #74087: Change "an uint" to "a uint"

Problem description:

Remainder from # 73404:

The affected documents contain text generated artefacts which contain the text "an uint".

Correct is "a uint".

The changes of the artefacts need changes in Metamodel and BSW UML Model.

Agreed solution:

Change "an uint" to "a uint" in metamodel artifacts.

–Last change on issue 74087 comment 2–

BW-C-Level:

Application	Specification	Bus
1	1	1