Document Title	SWS_TcpIp: 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_TcpIp

1.1 Specification Item ECUC_TcpIp_00160

Trace References:

none

Content:

Name	TcpIpIpV6ReassemblySegmentCount		
Description	Specifies the maximum number of consecutive data segments that can be managed in each reassembly buffer. If all fragments are received in order, only one segment will be needed. To deal with fragments received out of order this value should be configured bigger than 1.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	255 .. 1 1 .. 255		
Default value	5		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #76370: [TcpIp] min max range for TcpIpIpV6ReassemblySegmentCount

  Problem description:

  the range for the TcpIpIpV6ReassemblySegmentCount parameter is wrong

  <MAX>1</MAX>
  <MIN>255</MIN>

  Agreed solution:

  === EcuC ===
  SWS Item ECUC_TcpIp_00160
  Name TcpIpIpV6ReassemblySegmentCount
  Range 1 .. 255
  –Last change on issue 76370 comment 3–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.2 Specification Item SWS_TCPIP_00027

Trace References:

none

Content:

API function	Description
Dem_SetEventStatus	Called by SW-Cs or BSW modules to report monitor status information to the Dem. BSW modules calling Dem_SetEvent Status can safely ignore the return value.
Det_ReportRuntimeError	Service to report runtime errors. If a callout has been configured then this callout shall be called.
EthIf_GetPhysAddr	Obtains the physical source address used by the indexed controller
EthIf_ProvideTxBuffer	Provides access to a transmit buffer of the specified Ethernet controller.
EthIf_SetPhysAddr	Sets the physical source address used by the indexed controller.
EthIf_Transmit	Triggers transmission of a previously filled transmit buffer
EthSM_TcpIpModeIndication	This service is called by the TcpIp to report the actual TcpIp state (e.g. online, offline).

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #59085: Rollout of 'Runtime errors'

  Problem description:

  Inconsistencies in SWS with semantics of Default errors
  –Last change on issue 59085 comment 26–

  Agreed solution:

  solution in Column "G" of the new attachment https://www.autosar.org/bugzilla/attachment.cgi?id=4604

  Notes:
  - It is not enough just to migrate the error from one classification table to another. Please also check the related requirements (and background information) which is referring to that error and adapt them if needed.

- The review task of the ITs shall be done by the WP to which the specification "belongs".

*** BSW UML Model ***
SWS_CanNm:
———-
Chapter 8.6.1 Optional Interfaces:
Add within SWS_CanNm_00325 the API function Det_ReportRunTimeError

SWS_LinIf:
———-
SWS_LinIf_00359: add Det_ReportRuntimeError

SWS_UdpNm:
———-
Replace UDPNM_E_NO_INIT with UDPNM_E_UNINIT in description of API UdpNm_MainFunction_<Instance Id> (SWS_UdpNm_00234)

*** ECUC XML ***
Not affected. No configuration of runtime error reporting required (see SWS BSW General).
–Last change on issue 59085 comment 88–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.3 Specification Item SWS_TCPIP_00029

Trace References:

none

Content:

Service name:	TcpIp_RxIndicationTcpIp_RxIndication

Syntax:	void TcpIp_RxIndication(uint8 CtrlIdx, Eth_FrameType FrameType, boolean IsBroadcast, const uint8* PhysAddrPtr, const uint8* DataPtr, uint16 LenByte)	
Service ID[hex]:	0x14	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	CtrlIdxTcpIp_RxIndication.CtrlIdx	Index of the EthIf controller.
	FrameTypeTcpIp_RxIndication.Frame Type	frame type of received Ethernet frame
	IsBroadcastTcpIp_RxIndication.Is Broadcast	parameter to indicate a broadcast frame
	PhysAddrPtrTcpIp_RxIndication.Phys AddrPtr	pointer to Physical source address (MAC address in network byte order) of received Ethernet frame
	DataPtrTcpIp_RxIndication.DataPtr	Pointer to payload of the received Ethernet frame (i.e. Ethernet header is not provided).
	LenByteTcpIp_RxIndication.LenByte	Length of received data.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	By this API service the TCP/IP stack gets an indication and the data of a received frame.	

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

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

  Problem description:

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

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

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

  Agreed solution:

BSW UML model
————-

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

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

Introduce the following requirements prior to SRS_BSW_00371:

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

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

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

SRS_BSW_yyyyy: Input parameters of structure type shall be passed as a reference to a constant structure
Type: valid
Description: All input parameters of structure type shall be passed as a reference constant structure
Rationale: Passing input parameters of structure type by value would result in additional run-time overhead due to efforts for copying the whole structure.
Use case: For example a function named <Mip>_SomeFunction with a return type of Std_ReturnType and a single parameter named SomeParameter of type SomeStructure (where SomeStructure is a struct) is defined with the following signature:

Std_ReturnType <Mip>_SomeFunction(P2CONST(SomeStructure, AUTOMATIC, <MIP>_APPL_DATA) SomeParameter);
Dependencies: –
~~~~~~

Supporting Material: —

SRS_BSW_zzzzz: Input parameters of array type shall be passed as a reference to the constant array base type
Type: valid
Description: All input parameters of array type shall be passed as a reference to the constant array base type
Rationale: This effectively matches the behavior specified in the ISO-C:90 namely that a "declaration of a parameter as 'array of type' shall be adjusted to 'qualified pointer to
type'".
Use case: For example a function named <Mip>_SomeFunction with a return type of Std_ReturnType and a single parameter named SomeParameter of type array of uint8 is defined with the following signature:

Std_ReturnType <Mip>_SomeFunction(P2CONST(uint8, AUTOMATIC, <MIP>_APPL_DATA) SomeParameter);
Dependencies: –
Supporting Material: —

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

In SWS_Xfrm_00036 change

const <type>* dataElement

to

<paramtype> dataElement

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

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

In SWS_Xfrm_00038 change

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

to

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

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

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

The following paragraph shall then be removed:

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

In SWS_Xfrm_00040 change

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

to

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

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

<paramtype> is derived from <type> according to the parameter passing rules

rules defined by the SRS BSW General (see SRS_BSW_xxxxx, SRS_BSW_yyyyy, and SRS_BSW_zzzzz) and SWS BSW General (see SWS_BSW_00186 and SWS_BSW_00187).

In SWS_Xfrm_00044 change

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

to

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

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

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

The following paragraph shall then be removed:

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

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

In SWS_SomeIpXf_00138 change

const <type>* dataElement

to

<paramtype> dataElement
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

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

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

In SWS_SomeIpXf_00141 change

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

to

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

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

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

The following paragraph shall then be removed:

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

In SWS_SomeIpXf_00145 change

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

to

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

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

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

The following paragraph shall then be removed:

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

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

In SWS_ComXf_00007 change

const <type>* dataElement

to

<paramtype> dataElement

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

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

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

In SWS_EthTSyn_00040 make the parameter DataPtr of EthTSyn_RxIndication const.

Specification of SWS FlexRay Interface
~~~~~~

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

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

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

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

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

Specification of Dem
~~~~~~

no change required

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

~ [SWS_Efx_00307] Efx_RampGetSwitchPos: Include constant for pointer
~~~~~~~~~~~~~~~~~~~~~~

Input-parameter as like below.
boolean Efx_RampGetSwitchPos(const Efx_StateRamp_Type* State_cpst)

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

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

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

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

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

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

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

~ [SWS_Mfl_00223] Mfl_RampGetSwitchPos: Include constant for pointer Input-parameter as like below.
~~~~~~~~~~~~~~~~~~~~~

boolean Mfl_RampGetSwitchPos(const Mfl_StateRamp_Type* State_cpst)
–Last change on issue 68035 comment 135–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.4 Specification Item SWS_TCPIP_00040

Trace References:

SRS_Eth_00066

Content:

Service name:	TcpIp_DhcpReadOptionTcpIp_DhcpReadOption	
Syntax:	Std_ReturnType TcpIp_DhcpReadOption(TcpIp_LocalAddrIdType LocalIpAddrId, uint8 Option, uint8* DataLength, uint8* DataPtr)	
Service ID[hex]:	0x0D	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	LocalIpAddrIdTcpIp_DhcpRead Option.LocalIpAddrId	IP address identifier representing the local IP address and EthIf controller for which the DHCP option shall be read.
	OptionTcpIp_DhcpReadOption.Option	DHCP option according to IEFT RfC 2132, e.g. hostname
DataPtrTcpIp_DhcpReadOption.DataPtr	Pointer to memory containing DHCP option data	
Parameters (inout):	DataLengthTcpIp_DhcpRead Option.DataLength	As input parameter, contains the length of the provided data buffer. Will be overwritten with the length of the actual data.

None

Parameters (out):	DataPtrTcpIp_DhcpReadOption.DataPtr	Pointer to memory containing DHCP option data
Return value:	Std_ReturnType	Result of operation E_OK requested data retrieved successfully. E_NOT_OK requested data could not be retrieved.
Description:	By this API service the TCP/IP stack retrieves DHCP option data identified by parameter option for already received DHCP options.	

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

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

  Problem description:

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

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

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

  Agreed solution:

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

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

  Introduce the following requirements prior to SRS_BSW_00371:

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

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

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

SRS_BSW_yyyyy: Input parameters of structure type shall be passed as a reference to a constant structure
Type: valid
Description: All input parameters of structure type shall be passed as a reference constant structure
Rationale: Passing input parameters of structure type by value would result in additional run-time overhead due to efforts for copying the whole structure.
Use case: For example a function named <Mip>_SomeFunction with a return type of Std_ReturnType and a single parameter named SomeParameter of type SomeStructure (where SomeStructure is a struct) is defined with the following signature:

Std_ReturnType <Mip>_SomeFunction(P2CONST(SomeStructure, AUTOMATIC, <MIP>_APPL_DATA) SomeParameter);
Dependencies: –
Supporting Material: —

SRS_BSW_zzzzz: Input parameters of array type shall be passed as a reference to the constant array base type
Type: valid
Description: All input parameters of array type shall be passed as a reference to the constant array base type
Rationale: This effectively matches the behavior specified in the ISO-C:90 namely that a "declaration of a parameter as 'array of type' shall be adjusted to 'qualified pointer to
type'".
Use case: For example a function named <Mip>_SomeFunction with a return type of Std_ReturnType and a single parameter named SomeParameter of type array of uint8 is defined with the following signature:

Std_ReturnType <Mip>_SomeFunction(P2CONST(uint8, AUTOMATIC, <MIP>_APPL_DATA) SomeParameter);
Dependencies: –
Supporting Material: —

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

In SWS_Xfrm_00036 change
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

const <type>* dataElement

to

<paramtype> dataElement

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

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

In SWS_Xfrm_00038 change

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

to

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

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

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

The following paragraph shall then be removed:

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

In SWS_Xfrm_00040 change

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

to

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

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

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

In SWS_Xfrm_00044 change

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

to

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

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

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

The following paragraph shall then be removed:

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

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

In SWS_SomeIpXf_00138 change

const <type>* dataElement

to

<paramtype> dataElement

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

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

In SWS_SomeIpXf_00141 change

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

to

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

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

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

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

The following paragraph shall then be removed:

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

In SWS_SomeIpXf_00145 change

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

to

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

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

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

The following paragraph shall then be removed:

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

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

In SWS_ComXf_00007 change

const <type>* dataElement

to

<paramtype> dataElement

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

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

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

In SWS_EthTSyn_00040 make the parameter DataPtr of EthTSyn_RxIndication const.

Specification of SWS FlexRay Interface
~~~~~~

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Specification of EFX
~~~~~~

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

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

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

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

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

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

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

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

~ [SWS_Mfl_00285] Mfl_MedianSort_f32_f32: The parameter Array should be

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

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

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

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

BW-C-Level:

Application	Specification	Bus
1	4	1

1.5 Specification Item SWS_TCPIP_00042

Trace References:

none

Content:

The following table lists development errors that shall be distinguished by the TcpIp module:

Type or error	Relevance	Related error code	Value [hex]
API service called before initializing the module	Development	TCPIP_E_NOTINIT UNINIT	0x01
API service called with NULL pointer	Development	TCPIP_E_PARAM_POINTER	0x02
Invalid argument	Development	TCPIP_E_INV_ARG	0x03
No buffer space available	Development	TCPIP_E_NOBUFS	0x04
Message too long	Development	TCPIP_E_MSGSIZE	0x07
Protocol wrong type for socket	Development	TCPIP_E_PROTOTYPE	0x08

Type or error	Relevance	Related error code	Value [hex]
Address already in use	Development	TCPIP_E_ADDRINUSE	0x09
Can't assign requested address	Development	TCPIP_E_ADDRNOTAVAIL	0x0A
Socket is already connected	Development	TCPIP_E_ISCONN	0x0B
Socket is not connected	Development	TCPIP_E_NOTCONN	0x0C
Protocol not available	Development	TCPIP_E_NOPROTOOPT	0x0D
Address family not supported by protocol family	Development	TCPIP_E_AFNOSUPPORT	0x0E
Invalid configuration set selection	Development	TCPIP_E_INIT_FAILED	0x0F

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #59085: Rollout of 'Runtime errors'

  Problem description:

  Inconsistencies in SWS with semantics of Default errors
  –Last change on issue 59085 comment 26–

  Agreed solution:

  solution in Column "G" of the new attachment https://www.autosar.org/bugzilla/attachment.cgi?id=4604

  Notes:
  - It is not enough just to migrate the error from one classification table to another. Please also check the related requirements (and background information) which is referring to that error and adapt them if needed.
  - The review task of the ITs shall be done by the WP to which the specification "belongs".

  *** BSW UML Model ***
  SWS_CanNm:
  ———-
  Chapter 8.6.1 Optional Interfaces:
  Add within SWS_CanNm_00325 the API function Det_ReportRunTimeError

  SWS_LinIf:
  ———-
  SWS_LinIf_00359: add Det_ReportRuntimeError

  SWS_UdpNm:

———-

Replace UDPNM_E_NO_INIT with UDPNM_E_UNINIT in description of API UdpNm_MainFunction_<Instance Id> (SWS_UdpNm_00234)

*** ECUC XML ***
Not affected. No configuration of runtime error reporting required (see SWS BSW General).
–Last change on issue 59085 comment 88–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.6 Specification Item SWS_TCPIP_00065

Trace References:

none

Content:

Name:	TcpIp_IpAddrAssignmentTypeTcpIp_IpAddrAssignmentType	
Type:	Enumeration	
Range:	TCPIP_IPADDR_ASSIGNMENT_STATICTcp Ip_IpAddrAssignment Type.TCPIP_IPADDR_ASSIGNMENT_STATIC	Static configured IPv4/IPv6 address.
	TCPIP_IPADDR_ASSIGNMENT_LINKLOCAL_DOIPTcp Ip_IpAddrAssignment Type.TCPIP_IPADDR_ASSIGNMENT_LINKLOCAL_DOIP	Linklocal IPv4/IPv6 address assignment using DoIP parameters.
	TCPIP_IPADDR_ASSIGNMENT_DHCPTcp Ip_IpAddrAssignment Type.TCPIP_IPADDR_ASSIGNMENT_DHCP	Dynamic configured IPv4/IPv6 address by DHCP.
	TCPIP_IPADDR_ASSIGNMENT_LINKLOCALTcp Ip_IpAddrAssignment Type.TCPIP_IPADDR_ASSIGNMENT_LINKLOCAL	Linklocal IPv4/IPv6 address assignment.
	TCPIP_IPADDR_ASSIGNMENT_IPV6_ROUTERTcp Ip_IpAddrAssignment Type.TCPIP_IPADDR_ASSIGNMENT_IPV6_ROUTER	Dynamic configured IPv4/IPv6 address by Router Advertisement.
	TCPIP_IPADDR_ASSIGNMENT_ALLTcp Ip_IpAddrAssignment Type.TCPIP_IPADDR_ASSIGNMENT_ALL	All configured TcpIp AssignmentMethods with Tcp IpAssignmentTrigger set to TCPIP_MANUAL
Description:	Specification of IPv4/IPv6 address assignment policy.	

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #74847: [DoIP] How to configure DoIPUdpConnection with limited broadcast local IP address

  Problem description:

  according to ISO/FDIS 13400-2:2011 Vehicle Identification Request can be sent to multicast (limited broadcast). However, it is not specified in SWS how such a limited broadcast local address address can be configured for a DoIPUdpConnection.

  Since DoIP shall use SoAd_RequestIpAddrAssignment() with TCPIP_IPADDR_ASSIGNMENT_AUTOIP and TCPIP_IPADDR_ASSIGNMENT_DHCP, it is not possible to use a static local IP address.

  How shall limited broadcast local IP address be configured?
  –Last change on issue 74847 comment 2–

  Agreed solution:

  SWS DoIP
  ========

  Chapter 7 "Functional Specification"

  Change [SWS_DoIP_00204] from:
  If the DoIP Activation Line status changes from DOIP_ACTIVATION_LINE_INACTIVE to DOIP_ACTIVATION_LINE_ACTIVE, the DoIP module shall retrieve the SoConId of the first configured UDPConnection, via call to the SoAd_GetSoConId and trigger the IP Address assignment via 2 subsequent calls to SoAd_RequestIpAddrAssignment with the retrieved SoConId, LocalIpAddrPtr set to NULL_PTR and in the first call type set to TCPIP_IPADDR_ASSIGNMENT_LINKLOCAL_DOIP and in the second call type set to TCPIP_IPADDR_ASSIGNMENT_DHCP. (SRS_Eth_00081, SRS_Eth_00028)

  To:
  If the DoIP Activation Line status changes from DOIP_ACTIVATION_LINE_INACTIVE to DOIP_ACTIVATION_LINE_ACTIVE, the DoIP module shall loop over all DoIPTcpConnection, DoIPUdpConnection, and DoIPUdpVehicleAnnouncementConnections. For each of these DoIPConnections which has a DoIPRequestAddressAssignment set to true the DoIp module shall retrieve the corresponding SoConId via call to the SoAd_GetSoConId and trigger the IP Address assignment via subsequent calls to SoAd_RequestIpAddrAssignment with the retrieved SoConId, LocalIpAddrPtr and DefaultRouterPtr set to NULL_PTR, Netmask set to 0, and Type set to TCPIP_IPADDR_ASSIGNMENT_ALL. For each of these DoIPConnections (irrespective of the value of DoIPRequestAddressAssign-

ment) the DoIP module shall open the respective connection by an according call to SoAd_OpenSoCon. (SRS_Eth_00081, SRS_Eth_00028, SRS_Eth_00026).

Remove the note after SWS_DoIP_00204.

Remove SWS_DoIP_00003

Change [SWS_DoIP_00205] from:
If the function DoIP_SoConModeChg is called with Mode set to SOAD_SOCON_ONLINE, after SWS_DoIP_00003 has been performed, for a UDP vehicle announcement connection, the DoIP module shall send the vehicle announcement message via the corresponding Tx PDU configured in the DoIPUdpVehicleAnnouncementConnection and belonging to the reported socket connection.(SRS_Eth_00026)

To:
If the function DoIP_SoConModeChg is called with Mode set to SOAD_SOCON_ONLINE for a UDP vehicle announcement connection, the DoIP module shall send the vehicle announcement message via the corresponding Tx PDU configured in the DoIPUdpVehicleAnnouncementConnection and belonging to the reported socket connection.(SRS_Eth_00026)

Change [SWS_DoIP_00071] from:
If the DoIP module needs to send a vehicle announcement message because of SWS_DoIP_00003, it shall send the first vehicle announcement message via the configured DoIPUdpVehicleAnnouncementConnection after DoIPInitialVehicleAnnouncementTime as described in Table 6 and repeat this message DoIPVehicleAnnouncementRepetition times with a delay of DoIPVehicleAnnouncementInterval. The last "VIN/GID Status" byte of the Vehicle identification response message is optional as defined in the ISO13400-2 standard. It shall exist only if the "DoIPUseVehicleIdentificationSyncStatus" configuration parameter is set to True. (See SWS_DoIP_00086).(SRS_Eth_00026)

To:
If the DoIP module needs to send a vehicle announcement message (see SWS_DoIP_00205), it shall send the first vehicle announcement message via the configured DoIPUdpVehicleAnnouncementConnection after DoIPInitialVehicleAnnouncementTime as described in Table 6 and repeat this message DoIPVehicleAnnouncementRepetition times with a delay of DoIPVehicleAnnouncementInterval. The last "VIN/GID Status" byte of the Vehicle identification response message is optional as defined in the ISO13400-2 standard. It shall exist only if the "DoIPUseVehicleIdentificationSyncStatus" configuration parameter is set to True. (See

SWS_DoIP_00086).(SRS_Eth_00026)

Change [SWS_DoIP_00234] from:
If the DoIP Activation Line status changes from DOIP_ACTIVATION_LINE_ACTIVE to DOIP_ACTIVATION_LINE_INACTIVE, the DoIP module shall retrieve all the SoConId of all the configured UDPConnection, via call to the SoAd_GetSoConId and close all the UDP sockets by calls to the
SoAd_CloseSoCon with the all the retrieved SoConId. (SRS_Eth_00081, SRS_Eth_00028)

To:
If the DoIP Activation Line status changes from DOIP_ACTIVATION_LINE_ACTIVE to DOIP_ACTIVATION_LINE_INACTIVE, the DoIP module shall shall loop over all DoIPTcpConnection, DoIPUdpConnection, and DoIPUdpVehicleAnnouncementConnections. - For each of these DoIPConnections the DoIP module shall retrieve the corresponding SoConId via call to the SoAd_GetSoConId and close all the connection by a call to SoAd_CloseSoCon with the retrieved SoConId. (SRS_Eth_00081, SRS_Eth_00028)

Change [SWS_DoIP_00235] from:
When all UDP sockets are closed (i.e for all the Sockets the function DoIP_SoConModeChg was called with something else than SOAD_SOCON_ONLINE), the DoIP module shall retrieve the SoConId of the first configured UDPConnection, via call to the SoAd_GetSoConId and release the IP
Address assignment via the call to SoAd_ReleaseIpAddrAssignment with the retrieved SoConId. (SRS_Eth_00081, SRS_Eth_00028)

To:
In addition to SWS_DoIP_00234, the DoIP module shall release the corresponding IP Address assignment via the call to SoAd_ReleaseIpAddrAssignment for those connections that have DoIPRequestAddressAssignment set to true. (SRS_Eth_00081, SRS_Eth_00028)

Remove the note after SWS_DoIP_00235.

Chapter 9 "Sequence Diagrams"

Adapt sequence diagrams in section 9.4 "Activation Line Handling Active" and

in section 9.5 "Activation Line Handling Inactive" accordingly.

Chapter 10 "Configuration specification"

Add the following new config parameters to DoIPTcpConnection (10.2.13 DoIPTcpConnection), DoIPUdpConnection (10.2.14 DoIPUdpConnection), and DoIPUdpVehicleAnnouncementConnection (10.2.17 DoIPUdpVehicleAnnouncementConnection):

SWS Item ECUC_DoIP_xxxx1 :
Name DoIPRequestAddressAssignment
Description The DoIP module shall request IP address assignment by calling SoAd_RequestIpAddrAssignment() for the TcpIpLocalAddr related to this DoIpConnection.
Multiplicity 1
Type EcucBooleanParamDef
Default value true
Post-Build Variant Value false
Value Configuration Class Pre-compile time X All Variants
Link time –
Post-build time –
Scope / Dependency scope: local

SWS TcpIp
=========

Chapter 8 "API Specification"

Extend [SWS_TCPIP_00065] by a new enumeration literal:
* TCPIP_IPADDR_ASSIGNMENT_ALL - all configured TcpIpAssignmentMethods with TcpIpAssignmentTrigger set to TCPIP_MANUAL

Add an new specification item after [SWS_TCPIP_00080]:
[SWS_TCPIP_xxxx] In case TcpIp_RequestIpAddrAssignment() is called with parameter Type set to TCPIP_IPADDR_ASSIGNMENT_ALL, the IP address assignment for the IP address table entry specified by LocalAddId shall be initiated for all configured TcpIpAssignmentMethods with TcpIpAssignmentTrigger set to TCPIP_MANUAL.
–Last change on issue 74847 comment 42–

BW-C-Level:

Application	Specification	Bus
1	3	1

1.7 Specification Item SWS_TCPIP_00066

Trace References:

none

Content:

Name:	TcpIp_ReturnTypeTcpIp_ReturnType		
Type:	Enumeration		
Range:	TCPIP_E_OKTcpIp_Return Type.TCPIP_E_OK	–	operation completed successfully.
	TCPIP_E_NOT_OKTcp Ip_Return Type.TCPIP_E_NOT_OK	–	operation failed.
	TCPIP_E_PHYS_ADDR_MISSTcp Ip_Return Type.TCPIP_E_PHYS_ADDR_MISS	–	operation failed because of an ARP/NDP cache miss.
Description:	TcpIp specific return type.		

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #59085: Rollout of 'Runtime errors'

  Problem description:

  Inconsistencies in SWS with semantics of Default errors
  –Last change on issue 59085 comment 26–

  Agreed solution:

  solution in Column "G" of the new attachment https://www.autosar.org/bugzilla/attachment.cgi?id=4604

  Notes:
  - It is not enough just to migrate the error from one classification table to another. Please also check the related requirements (and background information) which is referring to that error and adapt them if needed.
  - The review task of the ITs shall be done by the WP to which the specification "belongs".

*** BSW UML Model ***
SWS_CanNm:
———-
Chapter 8.6.1 Optional Interfaces:
Add within SWS_CanNm_00325 the API function Det_ReportRunTimeError

SWS_LinIf:
———-
SWS_LinIf_00359: add Det_ReportRuntimeError

SWS_UdpNm:
———-
Replace UDPNM_E_NO_INIT with UDPNM_E_UNINIT in description of API UdpNm_MainFunction_<Instance Id> (SWS_UdpNm_00234)

*** ECUC XML ***
Not affected. No configuration of runtime error reporting required (see SWS BSW General).
–Last change on issue 59085 comment 88–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.8 Specification Item SWS_TCPIP_00119

Trace References:

none

Content:

The service TcpIp_ChangeParameter() shall change the parameter specified by ParameterId with the value (casted to the respective data type) specified by ParameterValue of the socket specified by for the SocketId.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #74376: [TcpIp] Parameter type in TcpIp_ChangeParameter undefined

  Problem description:

Currently, TcpIp_ChangeParameter has following definitions.

Std_ReturnType TcpIp_ChangeParameter(TcpIp_SocketIdType SocketId, TcpIp_ParamIdType ParameterId, uint8 *ParameterValue)

Based on TcpIp_ParamIdType ParameterId, different values (of different types) can be sent in uint8 *ParameterValue.

For example: If TcpIp_ParamIdType is TCPIP_PARAMID_TCP_NAGLE, a boolean need to be sent. But if TcpIp_ParamIdType is TCPIP_PARAMID_TCP_KEEPALIVE_TIME, then ParameterValue will carry address of different type (type casted to uint8 pointer).

Agreed solution:

~[SWS_TCPIP_00126]:
Name: TcpIp_ParamIdType
Range:
TCPIP_PARAMID_TCP_RXWND_MAX 0x00 Specifies the maximum TCP receive window for the socket. [uint16]
TCPIP_PARAMID_FRAMEPRIO 0x01 Specifies the frame priority for outgoing frames on the socket. [uint8]
TCPIP_PARAMID_TCP_NAGLE 0x02 Specifies if the Nagle Algorithm according to IETF RFC 896 is enabled or not. [boolean]
TCPIP_PARAMID_TCP_KEEPALIVE 0x03 Specifies if TCP Keep Alive Probes are sent on the socket connection. [boolean]
TCPIP_PARAMID_TTL 0x04 Specifies the time to live value for outgoing frames on the socket. For IPv6 this parameter specifies the value of the HopLimit field used in the IPv6 header. [uint8]
TCPIP_PARAMID_TCP_KEEPALIVE_TIME 0x05 Specifies the time in [s] between the last data packet sent (simple ACKs are not considered data) and the first keepalive probe. [uint32]
TCPIP_PARAMID_TCP_KEEPALIVE_PROBES_MAX 0x06 Specifies the maximum number of times that a keepalive probe is retransmitted. [uint16]
TCPIP_PARAMID_TCP_KEEPALIVE_INTERVAL 0x07 Specifies the interval in [s] between subsequent keepalive probes. [uint32]
TCPIP_PARAMID_TCP_OPTIONFILTER 0x08 Specifies which TCP option filter shall be applied on the related socket. [uint8]
TCPIP_PARAMID_PATHMTU_ENABLE 0x09 Specifies if the Path MTU Discovery shall be performed on the related socket. [boolean]
TCPIP_PARAMID_FLOWLABEL 0x0a The 20-bit Flow Label according to IETF RFC 6437. [uint32]
TCPIP_PARAMID_DSCP 0x0b The 6-bit Differentiated Service Code Point accord-

ing to IETF RFC 2474. [uint8]
TCPIP_PARAMID_UDP_CHECKSUM 0x0c Specifies if UDP checksum handling shall be enabled (TRUE) or skipped (FALSE) on the related socket. [boolean]
TCPIP_PARAMID_VENDOR_SPECIFIC 0x80 Start of vendor specific range of parameter IDs. [vendor specific]
Description:
Type for the specification of all supported Parameter IDs and their data types.

~[SWS_TCPIP_00119]
The service TcpIp_ChangeParameter() shall change the parameter specified by ParameterId with the value (casted to the respective data type) specified by ParameterValue for the SocketId.
–Last change on issue 74376 comment 13–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.9 Specification Item SWS_TCPIP_00126

Trace References:

none

Content:

Name:	TcpIp_ParamIdTypeTcpIp_ParamIdType
Type:	uint8

Range:	TCPIP_PARAMID_TCP_RXWND_MAXTcp Ip_ParamId Type.TCPIP_PARAMID_TCP_RXWND_MAX	0x00	Specifies the maximum TCP receive window for the socket. [uint16]
	TCPIP_PARAMID_FRAMEPRIOTcp Ip_ParamId Type.TCPIP_PARAMID_FRAMEPRIO	0x01	Specifies the frame priority for outgoing frames on the socket. [uint8]
	TCPIP_PARAMID_TCP_NAGLETcp Ip_ParamId Type.TCPIP_PARAMID_TCP_NAGLE	0x02	Specifies if the Nagle Algorithm according to IETF RFC 896 is enabled or not. [boolean]
	TCPIP_PARAMID_TCP_KEEPALIVETcp Ip_ParamId Type.TCPIP_PARAMID_TCP_KEEPALIVE	0x03	Specifies if TCP Keep Alive Probes are sent on the socket connection. [boolean]
	TCPIP_PARAMID_TTLTcp Ip_ParamId Type.TCPIP_PARAMID_TTL	0x04	Specifies the time to live value for outgoing frames on the socket. For IPv6 this parameter specifies the value of the HopLimit field used in the IPv6 header. [uint8]
	TCPIP_PARAMID_TCP_KEEPALIVE_TIMETcp Ip_ParamId Type.TCPIP_PARAMID_TCP_KEEPALIVE_TIME	0x05	Specifies the time in [s] between the last data packet sent (simple ACKs are not considered data) and the first keepalive probe. [uint32]
	TCPIP_PARAMID_TCP_KEEPALIVE_PROBES_MAXTcp Ip_ParamId Type.TCPIP_PARAMID_TCP_KEEPALIVE_PROBES_MAX	0x06	Specifies the maximum number of times that a keepalive probe is retransmitted. [uint16]
	TCPIP_PARAMID_TCP_KEEPALIVE_INTERVALTcp Ip_ParamId Type.TCPIP_PARAMID_TCP_KEEPALIVE_INTERVAL	0x07	Specifies the interval in [s] between subsequent keepalive probes. [uint32]
	TCPIP_PARAMID_TCP_OPTIONFILTERTcp Ip_ParamId Type.TCPIP_PARAMID_TCP_OPTIONFILTER	0x08	Specifies which TCP option filter shall be applied on the related socket. [uint8]
	TCPIP_PARAMID_PATHMTU_ENABLETcp Ip_ParamId Type.TCPIP_PARAMID_PATHMTU_ENABLE	0x09	Specifies if the Path MTU Discovery shall be performed on the related socket. [boolean]
	TCPIP_PARAMID_FLOWLABELTcp Ip_ParamId Type.TCPIP_PARAMID_FLOWLABEL	0x0a	The 20-bit Flow Label according to IETF RFC 6437. [uint32]
	TCPIP_PARAMID_DSCPTcp Ip_ParamId Type.TCPIP_PARAMID_DSCP	0x0b	The 6-bit Differentiated Service Code Point according to IETF RFC 2474. [uint8]
	TCPIP_PARAMID_UDP_CHECKSUMTcp Ip_ParamId Type.TCPIP_PARAMID_UDP_CHECKSUM	0x0c	0x0c Specifies if UDP checksum handling shall be enabled (TRUE) or skipped (FALSE) on the related socket. [boolean]
	TCPIP_PARAMID_VENDOR_SPECIFICTcp Ip_ParamId Type.TCPIP_PARAMID_VENDOR_SPECIFIC	0x80	Start of vendor specific range of parameter IDs. [vendor specific]
Description:	Type for the specification of all supported Parameter IDs and their data types.		

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #74376: [TcpIp] Parameter type in TcpIp_ChangeParameter undefined

  Problem description:

  Currently, TcpIp_ChangeParameter has following definitions.

  Std_ReturnType TcpIp_ChangeParameter(TcpIp_SocketIdType SocketId, TcpIp_ParamIdType ParameterId, uint8 *ParameterValue)

  Based on TcpIp_ParamIdType ParameterId, different values (of different types) can be sent in uint8 *ParameterValue.

  For example: If TcpIp_ParamIdType is TCPIP_PARAMID_TCP_NAGLE, a boolean need to be sent. But if TcpIp_ParamIdType is TCPIP_PARAMID_TCP_KEEPALIVE_TIME, then ParameterValue will carry address of different type (type casted to uint8 pointer).

  Agreed solution:

  ~[SWS_TCPIP_00126]:
  Name: TcpIp_ParamIdType
  Range:
  TCPIP_PARAMID_TCP_RXWND_MAX 0x00 Specifies the maximum TCP receive window for the socket. [uint16]
  TCPIP_PARAMID_FRAMEPRIO 0x01 Specifies the frame priority for outgoing frames on the socket. [uint8]
  TCPIP_PARAMID_TCP_NAGLE 0x02 Specifies if the Nagle Algorithm according to IETF RFC 896 is enabled or not. [boolean]
  TCPIP_PARAMID_TCP_KEEPALIVE 0x03 Specifies if TCP Keep Alive Probes are sent on the socket connection. [boolean]
  TCPIP_PARAMID_TTL 0x04 Specifies the time to live value for outgoing frames on the socket. For IPv6 this parameter specifies the value of the HopLimit field used in the IPv6 header. [uint8]
  TCPIP_PARAMID_TCP_KEEPALIVE_TIME 0x05 Specifies the time in [s] between the last data packet sent (simple ACKs are not considered data) and the first keepalive probe. [uint32]
  TCPIP_PARAMID_TCP_KEEPALIVE_PROBES_MAX 0x06 Specifies the maximum number of times that a keepalive probe is retransmitted. [uint16]
  TCPIP_PARAMID_TCP_KEEPALIVE_INTERVAL 0x07 Specifies the interval in [s] between subsequent keepalive probes. [uint32]
  TCPIP_PARAMID_TCP_OPTIONFILTER 0x08 Specifies which TCP option filter shall be applied on the related socket. [uint8]

TCPIP_PARAMID_PATHMTU_ENABLE 0x09 Specifies if the Path MTU Discovery shall be performed on the related socket. [boolean]
TCPIP_PARAMID_FLOWLABEL 0x0a The 20-bit Flow Label according to IETF RFC 6437. [uint32]
TCPIP_PARAMID_DSCP 0x0b The 6-bit Differentiated Service Code Point according to IETF RFC 2474. [uint8]
TCPIP_PARAMID_UDP_CHECKSUM 0x0c Specifies if UDP checksum handling shall be enabled (TRUE) or skipped (FALSE) on the related socket. [boolean]
TCPIP_PARAMID_VENDOR_SPECIFIC 0x80 Start of vendor specific range of parameter IDs. [vendor specific]
Description:
Type for the specification of all supported Parameter IDs and their data types.

~[SWS_TCPIP_00119]
The service TcpIp_ChangeParameter() shall change the parameter specified by ParameterId with the value (casted to the respective data type) specified by ParameterValue for the SocketId.
–Last change on issue 74376 comment 13–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.10 Specification Item SWS_TCPIP_00129

Trace References:

none

Content:

If development error detection is enabled and the parameter RemoteAddrPtr equals NULL_PTR, the TcpIp_TcpConnect function shall raise the development error code TCPIP_E_PARAM_POINTERand the TcpIp_TcpConnect function shall return E_NOT_OK.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #59085: Rollout of 'Runtime errors'

  Problem description:

Inconsistencies in SWS with semantics of Default errors
–Last change on issue 59085 comment 26–

Agreed solution:

solution in Column "G" of the new attachment https://www.autosar.org/bugzilla/attachment.cgi?id=4604

Notes:
- It is not enough just to migrate the error from one classification table to another. Please also check the related requirements (and background information) which is referring to that error and adapt them if needed.
- The review task of the ITs shall be done by the WP to which the specification "belongs".

*** BSW UML Model ***
SWS_CanNm:
———-
Chapter 8.6.1 Optional Interfaces:
Add within SWS_CanNm_00325 the API function Det_ReportRunTimeError

SWS_LinIf:
———-
SWS_LinIf_00359: add Det_ReportRuntimeError

SWS_UdpNm:
———-
Replace UDPNM_E_NO_INIT with UDPNM_E_UNINIT in description of API UdpNm_MainFunction_<Instance Id> (SWS_UdpNm_00234)

*** ECUC XML ***
Not affected. No configuration of runtime error reporting required (see SWS BSW General).
–Last change on issue 59085 comment 88–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.11 Specification Item SWS_TCPIP_00189

Trace References:

SRS_Eth_00066

Content:

Service name:	TcpIp_DhcpV6ReadOptionTcpIp_DhcpV6ReadOption	
Syntax:	Std_ReturnType TcpIp_DhcpV6ReadOption(TcpIp_LocalAddrIdType LocalIpAddrId, uint16 Option, uint16* DataLength, uint8* DataPtr)	
Service ID[hex]:	0x19	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	LocalIpAddrIdTcpIp_DhcpV6Read Option.LocalIpAddrId	IP address identifier representing the local IP address and EthIf controller for which the DHCPv6 option shall be read.
	OptionTcpIp_DhcpV6Read Option.Option	DHCP option according to IEFT RfC 3315, e.g. hostname
DataPtrTcpIp_DhcpV6ReadOption.Data Ptr	Pointer to memory containing DHCPv6 option data	
Parameters (inout):	DataLengthTcpIp_DhcpV6Read Option.DataLength	As input parameter, contains the length of the provided data buffer. Will be overwritten with the length of the actual data.

None

Parameters (out):	DataPtrTcpIp_DhcpV6ReadOption.Data Ptr	Pointer to memory containing DHCPv6 option data
Return value:	Std_ReturnType	Result of operation E_OK: requested data retrieved successfully. E_NOT_OK: requested data could not be retrieved.
Description:	By this API service the TCP/IP stack retrieves DHCPv6 option data identified by parameter option for already received DHCPv6 options.	

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

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

  Problem description:

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

  In addition to that there shall be a SWS item that states that input parameters

of integral and enum type shall be passed by value whereas input parameters of structure type shall be passed by reference.

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

Agreed solution:

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

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

Introduce the following requirements prior to SRS_BSW_00371:

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

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

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

SRS_BSW_yyyyy: Input parameters of structure type shall be passed as a reference to a constant structure
Type: valid
Description: All input parameters of structure type shall be passed as a reference constant structure
Rationale: Passing input parameters of structure type by value would result in additional run-time overhead due to efforts for copying the whole structure.
~~~~~~

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

Std_ReturnType <Mip>_SomeFunction(P2CONST(SomeStructure, AUTOMATIC, <MIP>_APPL_DATA) SomeParameter);
Dependencies: –
Supporting Material: —

SRS_BSW_zzzzz: Input parameters of array type shall be passed as a reference to the constant array base type
Type: valid
Description: All input parameters of array type shall be passed as a reference to the constant array base type
Rationale: This effectively matches the behavior specified in the ISO-C:90 namely that a "declaration of a parameter as 'array of type' shall be adjusted to 'qualified pointer to
type'".
Use case: For example a function named <Mip>_SomeFunction with a return type of Std_ReturnType and a single parameter named SomeParameter of type array of uint8 is defined with the following signature:

Std_ReturnType <Mip>_SomeFunction(P2CONST(uint8, AUTOMATIC, <MIP>_APPL_DATA) SomeParameter);
Dependencies: –
Supporting Material: —

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

In SWS_Xfrm_00036 change

const <type>* dataElement

to

<paramtype> dataElement

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

"

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

In SWS_Xfrm_00038 change

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

to

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

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

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

The following paragraph shall then be removed:

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

In SWS_Xfrm_00040 change

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

to

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

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

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

In SWS_Xfrm_00044 change

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

to

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

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

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

The following paragraph shall then be removed:

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

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

In SWS_SomeIpXf_00138 change

const <type>* dataElement

to

<paramtype> dataElement

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

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

In SWS_SomeIpXf_00141 change

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

to

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

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

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

The following paragraph shall then be removed:

For the arguments of ClientServerOperation which are handed over to the transformer as data_1, ..., data_n the requirements to API parameters stated in

chapter API Parameters of [5, SWS RTE] are valid (especially [SWS_Rte_01017], [SWS_Rte_01018] and [SWS_Rte_05107]).

In SWS_SomeIpXf_00145 change

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

to

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

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

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

The following paragraph shall then be removed:

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

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

In SWS_ComXf_00007 change

const <type>* dataElement

to

<paramtype> dataElement
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

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

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

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

In SWS_EthTSyn_00040 make the parameter DataPtr of EthTSyn_RxIndication const.

Specification of SWS FlexRay Interface
~~~~~~

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

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

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

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

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

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

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

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

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

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

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

Specification of EthIf
~~~~~~
~~~~~~

no change required

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

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

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

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

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

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

Specification of J1939Nm
~~~~~~

Change parameter 'name' of User_AddressClaimedIndication to type 'const uint8*'

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

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

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

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

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

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

~ [SWS_Efx_00376] Efx_MedianSort: The parameter <InType>* Array should be InOut instead of In parameter as like below.

Parameters (in): N Size of an array
Parameters (inout): Array Pointer to an array

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

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

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

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

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

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

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

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

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

Parameters (out): State_cpst Pointer to internal state structure

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

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

BW-C-Level:

Application	Specification	Bus
1	4	1

1.12 Specification Item SWS_TCPIP_00223

Trace References:

SRS_Eth_00103

Content:

Service name:	<Up_RxIndication><Up_RxIndication>	
Syntax:	void <Up_RxIndication>(TcpIp_SocketIdType SocketId, const TcpIp_SockAddrType* RemoteAddrPtr, const uint8* BufPtr, uint16 Length)	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different SocketIds. Non reentrant for the same SocketId.	
Parameters (in):	SocketId<Up_RxIndication>.SocketId	Socket identifier of the related local socket resource.
	RemoteAddrPtr<Up_Rx Indication>.RemoteAddrPtr	Pointer to memory containing IP address and port of the remote host which sent the data.
	BufPtr<Up_RxIndication>.BufPtr	Pointer to the received data.
	Length<Up_RxIndication>.Length	Data length of the received TCP segment or UDP datagram.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	The TCP/IP stack calls this primitive after the reception of data on a socket. The socket identifier along with configuration information determines which module is to be called.	

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

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

  Problem description:

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

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

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

  Agreed solution:

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

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

  Introduce the following requirements prior to SRS_BSW_00371:

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

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

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

Supporting Material: —

SRS_BSW_yyyyy: Input parameters of structure type shall be passed as a reference to a constant structure
Type: valid
Description: All input parameters of structure type shall be passed as a reference constant structure
Rationale: Passing input parameters of structure type by value would result in additional run-time overhead due to efforts for copying the whole structure.
Use case: For example a function named <Mip>_SomeFunction with a return type of Std_ReturnType and a single parameter named SomeParameter of type SomeStructure (where SomeStructure is a struct) is defined with the following signature:

Std_ReturnType <Mip>_SomeFunction(P2CONST(SomeStructure, AUTOMATIC, <MIP>_APPL_DATA) SomeParameter);
Dependencies: –
Supporting Material: —

SRS_BSW_zzzzz: Input parameters of array type shall be passed as a reference to the constant array base type
Type: valid
Description: All input parameters of array type shall be passed as a reference to the constant array base type
Rationale: This effectively matches the behavior specified in the ISO-C:90 namely that a "declaration of a parameter as 'array of type' shall be adjusted to 'qualified pointer to
type'".
Use case: For example a function named <Mip>_SomeFunction with a return type of Std_ReturnType and a single parameter named SomeParameter of type array of uint8 is defined with the following signature:

Std_ReturnType <Mip>_SomeFunction(P2CONST(uint8, AUTOMATIC, <MIP>_APPL_DATA) SomeParameter);
Dependencies: –
Supporting Material: —

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

In SWS_Xfrm_00036 change

const <type>* dataElement

to

<paramtype> dataElement

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

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

In SWS_Xfrm_00038 change

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

to

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

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

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

The following paragraph shall then be removed:

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

[SWS_Rte_01018] and [SWS_Rte_05107]).

In SWS_Xfrm_00040 change

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

to

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

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

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

In SWS_Xfrm_00044 change

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

to

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

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

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

The following paragraph shall then be removed:

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

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

In SWS_SomeIpXf_00138 change

const <type>* dataElement

to

<paramtype> dataElement

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

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

In SWS_SomeIpXf_00141 change

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

to

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

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

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

The following paragraph shall then be removed:

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

In SWS_SomeIpXf_00145 change

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

to

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

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

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

The following paragraph shall then be removed:

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

Specification of COM Based Transformer

~~~~~~

In SWS_ComXf_00007 change

const <type>* dataElement

to

<paramtype> dataElement

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

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

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

In SWS_EthTSyn_00040 make the parameter DataPtr of EthTSyn_RxIndication const.

Specification of SWS FlexRay Interface
~~~~~~

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

Specification of ADC
~~~~~~

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

ters (in) to Parameters (out)

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

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

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

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

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

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

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

Specification of E2ELibrary
~~~~~~

no change required

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Specification of EFX
~~~~~~~~~~~~~~~~~~~~~

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

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

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

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

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

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

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

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

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

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

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

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

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

BW-C-Level:

Application	Specification	Bus
1	4	1

1.13 Specification Item SWS_TCPIP_00228

Trace References:

SRS_Eth_00103

Content:

Service name:	<Up_CopyTxData><Up_CopyTxData>
Syntax:	BufReq_ReturnType <Up_CopyTxData>(TcpIp_SocketIdType SocketId, uint8* BufPtr, uint16 BufLength)
Sync/Async:	Synchronous
Reentrancy:	Reentrant for different SocketIds. Non reentrant for the same SocketId.

Parameters (in):	SocketId<Up_CopyTxData>.SocketId	Socket identifier of the related local socket resource.
	BufPtr<Up_CopyTxData>.BufPtr	Pointer to buffer for transmission data.
	BufLength<Up_CopyTxData>.BufLength	Length of provided data buffer.
Parameters (inout):	None	

None

Parameters (out):	BufPtr<Up_CopyTxData>.BufPtr	Pointer to buffer for transmission data.
Return value:	BufReq_ReturnType	BUFREQ_OK: Data has been copied to the transmit buffer completely as requested. BUFREQ_E_NOT_OK: Data has not been copied. Request failed. (No further action for TcpIp required. Later the upper layer might either close the socket or retry the transmit request)
Description:	This service requests to copy data for transmission to the buffer indicated. This call is triggered by TcpIp_Transmit(). Note: The call to <Up>_CopyTxData() may happen in the context of TcpIp_Transmit().	

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

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

  Problem description:

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

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

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

  Agreed solution:

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

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

Introduce the following requirements prior to SRS_BSW_00371:

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

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

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

SRS_BSW_yyyyy: Input parameters of structure type shall be passed as a reference to a constant structure
Type: valid
Description: All input parameters of structure type shall be passed as a reference constant structure
Rationale: Passing input parameters of structure type by value would result in additional run-time overhead due to efforts for copying the whole structure.
Use case: For example a function named <Mip>_SomeFunction with a return type of Std_ReturnType and a single parameter named SomeParameter of type SomeStructure (where SomeStructure is a struct) is defined with the following signature:

Std_ReturnType <Mip>_SomeFunction(P2CONST(SomeStructure, AUTOMATIC, <MIP>_APPL_DATA) SomeParameter);
Dependencies: –
Supporting Material: —

SRS_BSW_zzzzz: Input parameters of array type shall be passed as a reference to the constant array base type
Type: valid
Description: All input parameters of array type shall be passed as a reference to the
~~~~~~

constant array base type
Rationale: This effectively matches the behavior specified in the ISO-C:90 namely that a "declaration of a parameter as 'array of type' shall be adjusted to 'qualified pointer to
type'".
Use case: For example a function named <Mip>_SomeFunction with a return type of Std_ReturnType and a single parameter named SomeParameter of type array of uint8 is defined with the following signature:

Std_ReturnType <Mip>_SomeFunction(P2CONST(uint8, AUTOMATIC, <MIP>_APPL_DATA) SomeParameter);
Dependencies: –
Supporting Material: —

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

In SWS_Xfrm_00036 change

const <type>* dataElement

to

<paramtype> dataElement

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

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

In SWS_Xfrm_00038 change

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

to
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

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

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

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

The following paragraph shall then be removed:

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

In SWS_Xfrm_00040 change

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

to

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

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

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

In SWS_Xfrm_00044 change

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

to

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

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

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

The following paragraph shall then be removed:

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

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

In SWS_SomeIpXf_00138 change

const <type>* dataElement

to

<paramtype> dataElement

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

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

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

In SWS_SomeIpXf_00141 change

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

to

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

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

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

The following paragraph shall then be removed:

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

In SWS_SomeIpXf_00145 change

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

to

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

and add the following to the where clause after the API table after the bullet

"type is data type of the data element
"

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

The following paragraph shall then be removed:

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

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

In SWS_ComXf_00007 change

const <type>* dataElement

to

<paramtype> dataElement

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

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

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

In SWS_EthTSyn_00040 make the parameter DataPtr of EthTSyn_RxIndication const.

Specification of SWS FlexRay Interface
~~~~~~

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Specification of LdCom

~~~~~~

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

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

Specification of Lin
~~~~~~

PduInfoPtr needs to be const in Std_ReturnType Lin_SendFrame(uint8 Channel, const Lin_PduType* PduInfoPtr)

Specification of PduR
~~~~~~

* PduR_<User:LoTp>CopyTxData
add const to "RetryInfoType* retry"

Specification of J1939Nm
~~~~~~

Change parameter ’name’ of User_AddressClaimedIndication to type ’const uint8*’

Specification of SoAd
~~~~~~

=> everything already fixed with RfC 65633

Specification of SPIHandlerDriver
~~~~~~

==> nothing to change for SWS SPI

Specification of SynchronizedTimeBaseManager
~~~~~~
~~~~~~

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

BW-C-Level:

Application	Specification	Bus
1	4	1
~~~~~~~~~~~~~~~~~~~~~~

1.14 Specification Item SWS_TCPIP_00233

Trace References:

SRS_Eth_00066

Content:

If development error detection is enabled: TcpIp_DhcpReadOption() shall check if the parameter LocalIpAddrId is valid. If the check fails, TcpIp_DhcpReadOption() shall raise the development error TCPIP_E_INV_ARGand return E_NOT_OK.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #59085: Rollout of 'Runtime errors'

  Problem description:

  Inconsistencies in SWS with semantics of Default errors
  –Last change on issue 59085 comment 26–

  Agreed solution:

  solution in Column "G" of the new attachment https://www.autosar.org/bugzilla/attachment.cgi?id=4604

  Notes:
  - It is not enough just to migrate the error from one classification table to another. Please also check the related requirements (and background information) which is referring to that error and adapt them if needed.
  - The review task of the ITs shall be done by the WP to which the specification "belongs".

  *** BSW UML Model ***
  SWS_CanNm:
  ———-
  Chapter 8.6.1 Optional Interfaces:
  Add within SWS_CanNm_00325 the API function Det_ReportRunTimeError

  SWS_LinIf:
  ———-
  SWS_LinIf_00359: add Det_ReportRuntimeError

  SWS_UdpNm:
  ———-

Replace UDPNM_E_NO_INIT with UDPNM_E_UNINIT in description of API UdpNm_MainFunction_<Instance Id> (SWS_UdpNm_00234)

*** ECUC XML ***
Not affected. No configuration of runtime error reporting required (see SWS BSW General).
–Last change on issue 59085 comment 88–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.15 Specification Item SWS_TCPIP_00234

Trace References:

SRS_Eth_00066

Content:

If development error detection is enabled: TcpIp_DhcpReadOption() shall check if the parameter Option is valid. If the check fails, TcpIp_DhcpReadOption() shall raise the development error TCPIP_E_INV_ARGand return E_NOT_OK.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #59085: Rollout of 'Runtime errors'

  Problem description:

  Inconsistencies in SWS with semantics of Default errors
  –Last change on issue 59085 comment 26–

  Agreed solution:

  solution in Column "G" of the new attachment https://www.autosar.org/bugzilla/attachment.cgi?id=4604

  Notes:
  - It is not enough just to migrate the error from one classification table to another. Please also check the related requirements (and background information) which is referring to that error and adapt them if needed.
  - The review task of the ITs shall be done by the WP to which the specification "belongs".

*** BSW UML Model ***
SWS_CanNm:
———-
Chapter 8.6.1 Optional Interfaces:
Add within SWS_CanNm_00325 the API function Det_ReportRunTimeError

SWS_LinIf:
———-
SWS_LinIf_00359: add Det_ReportRuntimeError

SWS_UdpNm:
———-
Replace UDPNM_E_NO_INIT with UDPNM_E_UNINIT in description of API UdpNm_MainFunction_<Instance Id> (SWS_UdpNm_00234)

*** ECUC XML ***
Not affected. No configuration of runtime error reporting required (see SWS BSW General).
–Last change on issue 59085 comment 88–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.16 Specification Item SWS_TCPIP_00235

Trace References:

SRS_Eth_00066

Content:

If development error detection is enabled: TcpIp_DhcpReadOption() shall check if the parameter DataLength is valid (i.e. the buffer is large enough for the requested option). If the check fails, TcpIp_DhcpReadOption() shall raise the development error TCPIP_E_INV_ARGand return E_NOT_OK.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #59085: Rollout of 'Runtime errors'

Problem description:

Inconsistencies in SWS with semantics of Default errors
–Last change on issue 59085 comment 26–

Agreed solution:

solution in Column "G" of the new attachment https://www.autosar.org/bugzilla/attachment.cgi?id=4604

Notes:
- It is not enough just to migrate the error from one classification table to another. Please also check the related requirements (and background information) which is referring to that error and adapt them if needed.
- The review task of the ITs shall be done by the WP to which the specification "belongs".

*** BSW UML Model ***
SWS_CanNm:
———-
Chapter 8.6.1 Optional Interfaces:
Add within SWS_CanNm_00325 the API function Det_ReportRunTimeError

SWS_LinIf:
———-
SWS_LinIf_00359: add Det_ReportRuntimeError

SWS_UdpNm:
———-
Replace UDPNM_E_NO_INIT with UDPNM_E_UNINIT in description of API UdpNm_MainFunction_<Instance Id> (SWS_UdpNm_00234)

*** ECUC XML ***
Not affected. No configuration of runtime error reporting required (see SWS BSW General).
–Last change on issue 59085 comment 88–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.17 Specification Item SWS_TCPIP_00238

Trace References:

SRS_Eth_00066

Content:

If development error detection is enabled: TcpIp_DhcpV6ReadOption() shall check if the parameter LocalIpAddrId is valid. If the check fails, TcpIp_DhcpV6ReadOption() shall raise the development error TCPIP_E_INV_ARGand return E_NOT_OK.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #59085: Rollout of 'Runtime errors'

  Problem description:

  Inconsistencies in SWS with semantics of Default errors
  –Last change on issue 59085 comment 26–

  Agreed solution:

  solution in Column "G" of the new attachment https://www.autosar.org/bugzilla/attachment.cgi?id=4604

  Notes:
  - It is not enough just to migrate the error from one classification table to another. Please also check the related requirements (and background information) which is referring to that error and adapt them if needed.
  - The review task of the ITs shall be done by the WP to which the specification "belongs".

  *** BSW UML Model ***
  SWS_CanNm:
  ———-
  Chapter 8.6.1 Optional Interfaces:
  Add within SWS_CanNm_00325 the API function Det_ReportRunTimeError

  SWS_LinIf:
  ———-
  SWS_LinIf_00359: add Det_ReportRuntimeError

  SWS_UdpNm:
  ———-

Replace UDPNM_E_NO_INIT with UDPNM_E_UNINIT in description of API UdpNm_MainFunction_<Instance Id> (SWS_UdpNm_00234)

*** ECUC XML ***
Not affected. No configuration of runtime error reporting required (see SWS BSW General).
–Last change on issue 59085 comment 88–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.18 Specification Item SWS_TCPIP_00239

Trace References:

SRS_Eth_00066

Content:

If development error detection is enabled: TcpIp_DhcpV6ReadOption() shall check if the parameter Option is valid. If the check fails, TcpIp_DhcpV6ReadOption() shall raise the development error TCPIP_E_INV_ARGand return E_NOT_OK.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #59085: Rollout of 'Runtime errors'

  Problem description:

  Inconsistencies in SWS with semantics of Default errors
  –Last change on issue 59085 comment 26–

  Agreed solution:

  solution in Column "G" of the new attachment https://www.autosar.org/bugzilla/attachment.cgi?id=4604

  Notes:
  - It is not enough just to migrate the error from one classification table to another. Please also check the related requirements (and background information) which is referring to that error and adapt them if needed.
  - The review task of the ITs shall be done by the WP to which the specification "belongs".

*** BSW UML Model ***
SWS_CanNm:
———-
Chapter 8.6.1 Optional Interfaces:
Add within SWS_CanNm_00325 the API function Det_ReportRunTimeError

SWS_LinIf:
———-
SWS_LinIf_00359: add Det_ReportRuntimeError

SWS_UdpNm:
———-
Replace UDPNM_E_NO_INIT with UDPNM_E_UNINIT in description of API UdpNm_MainFunction_<Instance Id> (SWS_UdpNm_00234)

*** ECUC XML ***
Not affected. No configuration of runtime error reporting required (see SWS BSW General).
–Last change on issue 59085 comment 88–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.19 Specification Item SWS_TCPIP_00240

Trace References:

SRS_Eth_00066

Content:

If development error detection is enabled: TcpIp_DhcpV6ReadOption() shall check if the parameter DataLength is valid (i.e. the buffer is large enough for the requested option). If the check fails, TcpIp_DhcpV6ReadOption() shall raise the development error TCPIP_E_INV_ARGand return E_NOT_OK.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #59085: Rollout of 'Runtime errors'

Problem description:

Inconsistencies in SWS with semantics of Default errors
–Last change on issue 59085 comment 26–

Agreed solution:

solution in Column "G" of the new attachment https://www.autosar.org/bugzilla/attachment.cgi?id=4604

Notes:
- It is not enough just to migrate the error from one classification table to another. Please also check the related requirements (and background information) which is referring to that error and adapt them if needed.
- The review task of the ITs shall be done by the WP to which the specification "belongs".

*** BSW UML Model ***
SWS_CanNm:
———-
Chapter 8.6.1 Optional Interfaces:
Add within SWS_CanNm_00325 the API function Det_ReportRunTimeError

SWS_LinIf:
———-
SWS_LinIf_00359: add Det_ReportRuntimeError

SWS_UdpNm:
———-
Replace UDPNM_E_NO_INIT with UDPNM_E_UNINIT in description of API UdpNm_MainFunction_<Instance Id> (SWS_UdpNm_00234)

*** ECUC XML ***
Not affected. No configuration of runtime error reporting required (see SWS BSW General).
–Last change on issue 59085 comment 88–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.20 Specification Item SWS_TCPIP_00243

Trace References:

SRS_Eth_00065

Content:

If development error detection is enabled: TcpIp_DhcpWriteOption() shall check if the parameter LocalIpAddrId is valid. If the check fails, TcpIp_DhcpWriteOption() shall raise the development error TCPIP_E_INV_ARGand return E_NOT_OK.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #59085: Rollout of 'Runtime errors'

  Problem description:

  Inconsistencies in SWS with semantics of Default errors
  –Last change on issue 59085 comment 26–

  Agreed solution:

  solution in Column "G" of the new attachment https://www.autosar.org/bugzilla/attachment.cgi?id=4604

  Notes:
  - It is not enough just to migrate the error from one classification table to another. Please also check the related requirements (and background information) which is referring to that error and adapt them if needed.
  - The review task of the ITs shall be done by the WP to which the specification "belongs".

  *** BSW UML Model ***
  SWS_CanNm:
  ———-
  Chapter 8.6.1 Optional Interfaces:
  Add within SWS_CanNm_00325 the API function Det_ReportRunTimeError

  SWS_LinIf:
  ———-
  SWS_LinIf_00359: add Det_ReportRuntimeError

  SWS_UdpNm:
  ———-

Replace UDPNM_E_NO_INIT with UDPNM_E_UNINIT in description of API UdpNm_MainFunction_<Instance Id> (SWS_UdpNm_00234)

*** ECUC XML ***
Not affected. No configuration of runtime error reporting required (see SWS BSW General).
–Last change on issue 59085 comment 88–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.21 Specification Item SWS_TCPIP_00244

Trace References:

SRS_Eth_00065

Content:

If development error detection is enabled: TcpIp_DhcpWriteOption() shall check if the parameter Option is valid. If the check fails, TcpIp_DhcpWriteOption() shall raise the development error TCPIP_E_INV_ARGand return E_NOT_OK.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #59085: Rollout of 'Runtime errors'

  Problem description:

  Inconsistencies in SWS with semantics of Default errors
  –Last change on issue 59085 comment 26–

  Agreed solution:

  solution in Column "G" of the new attachment https://www.autosar.org/bugzilla/attachment.cgi?id=4604

  Notes:
  - It is not enough just to migrate the error from one classification table to another. Please also check the related requirements (and background information) which is referring to that error and adapt them if needed.
  - The review task of the ITs shall be done by the WP to which the specification "belongs".

*** BSW UML Model ***
SWS_CanNm:
———-
Chapter 8.6.1 Optional Interfaces:
Add within SWS_CanNm_00325 the API function Det_ReportRunTimeError

SWS_LinIf:
———-
SWS_LinIf_00359: add Det_ReportRuntimeError

SWS_UdpNm:
———-
Replace UDPNM_E_NO_INIT with UDPNM_E_UNINIT in description of API UdpNm_MainFunction_<Instance Id> (SWS_UdpNm_00234)

*** ECUC XML ***
Not affected. No configuration of runtime error reporting required (see SWS BSW General).
–Last change on issue 59085 comment 88–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.22 Specification Item SWS_TCPIP_00245

Trace References:

SRS_Eth_00065

Content:

If development error detection is enabled: TcpIp_DhcpWriteOption() shall check if the parameter DataLength is valid (i.e. the length of the provided option is not larger than supported by the protocol). If the check fails, TcpIp_DhcpWriteOption() shall raise the development error TCPIP_E_INV_ARGand return E_NOT_OK.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #59085: Rollout of 'Runtime errors'

Problem description:

Inconsistencies in SWS with semantics of Default errors
–Last change on issue 59085 comment 26–

Agreed solution:

solution in Column "G" of the new attachment https://www.autosar.org/bugzilla/attachment.cgi?id=4604

Notes:
- It is not enough just to migrate the error from one classification table to another. Please also check the related requirements (and background information) which is referring to that error and adapt them if needed.
- The review task of the ITs shall be done by the WP to which the specification "belongs".

*** BSW UML Model ***
SWS_CanNm:
———-
Chapter 8.6.1 Optional Interfaces:
Add within SWS_CanNm_00325 the API function Det_ReportRunTimeError

SWS_LinIf:
———-
SWS_LinIf_00359: add Det_ReportRuntimeError

SWS_UdpNm:
———-
Replace UDPNM_E_NO_INIT with UDPNM_E_UNINIT in description of API UdpNm_MainFunction_<Instance Id> (SWS_UdpNm_00234)

*** ECUC XML ***
Not affected. No configuration of runtime error reporting required (see SWS BSW General).
–Last change on issue 59085 comment 88–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.23 Specification Item SWS_TCPIP_00248

Trace References:

SRS_Eth_00065

Content:

If development error detection is enabled: TcpIp_DhcpV6WriteOption() shall check if the parameter LocalIpAddrId is valid. If the check fails, TcpIp_DhcpV6WriteOption() shall raise the development error TCPIP_E_INV_ARGand return E_NOT_OK.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #59085: Rollout of 'Runtime errors'

  Problem description:

  Inconsistencies in SWS with semantics of Default errors
  –Last change on issue 59085 comment 26–

  Agreed solution:

  solution in Column "G" of the new attachment https://www.autosar.org/bugzilla/attachment.cgi?id=4604

  Notes:
  - It is not enough just to migrate the error from one classification table to another. Please also check the related requirements (and background information) which is referring to that error and adapt them if needed.
  - The review task of the ITs shall be done by the WP to which the specification "belongs".

  *** BSW UML Model ***
  SWS_CanNm:
  ———-
  Chapter 8.6.1 Optional Interfaces:
  Add within SWS_CanNm_00325 the API function Det_ReportRunTimeError

  SWS_LinIf:
  ———-
  SWS_LinIf_00359: add Det_ReportRuntimeError

  SWS_UdpNm:
  ———-

Replace UDPNM_E_NO_INIT with UDPNM_E_UNINIT in description of API UdpNm_MainFunction_<Instance Id> (SWS_UdpNm_00234)

*** ECUC XML ***
Not affected. No configuration of runtime error reporting required (see SWS BSW General).
–Last change on issue 59085 comment 88–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.24 Specification Item SWS_TCPIP_00249

Trace References:

SRS_Eth_00065

Content:

If development error detection is enabled: TcpIp_DhcpV6WriteOption() shall check if the parameter Option is valid. If the check fails, TcpIp_DhcpV6WriteOption() shall raise the development error TCPIP_E_INV_ARGand return E_NOT_OK.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #59085: Rollout of 'Runtime errors'

  Problem description:

  Inconsistencies in SWS with semantics of Default errors
  –Last change on issue 59085 comment 26–

  Agreed solution:

  solution in Column "G" of the new attachment https://www.autosar.org/bugzilla/attachment.cgi?id=4604

  Notes:
  - It is not enough just to migrate the error from one classification table to another. Please also check the related requirements (and background information) which is referring to that error and adapt them if needed.
  - The review task of the ITs shall be done by the WP to which the specification "belongs".

*** BSW UML Model ***
SWS_CanNm:
———-
Chapter 8.6.1 Optional Interfaces:
Add within SWS_CanNm_00325 the API function Det_ReportRunTimeError

SWS_LinIf:
———-
SWS_LinIf_00359: add Det_ReportRuntimeError

SWS_UdpNm:
———-
Replace UDPNM_E_NO_INIT with UDPNM_E_UNINIT in description of API UdpNm_MainFunction_<Instance Id> (SWS_UdpNm_00234)

*** ECUC XML ***
Not affected. No configuration of runtime error reporting required (see SWS BSW General).
–Last change on issue 59085 comment 88–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.25 Specification Item SWS_TCPIP_00250

Trace References:

SRS_Eth_00065

Content:

If development error detection is enabled: TcpIp_DhcpV6WriteOption() shall check if the parameter DataLength is valid (i.e. the length of the provided option is not larger than supported by the protocol). If the check fails, TcpIp_DhcpV6WriteOption() shall raise the development error TCPIP_E_INV_ARGand return E_NOT_OK.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #59085: Rollout of 'Runtime errors'

Problem description:

Inconsistencies in SWS with semantics of Default errors
–Last change on issue 59085 comment 26–

Agreed solution:

solution in Column "G" of the new attachment https://www.autosar.org/bugzilla/attachment.cgi?id=4604

Notes:
- It is not enough just to migrate the error from one classification table to another. Please also check the related requirements (and background information) which is referring to that error and adapt them if needed.
- The review task of the ITs shall be done by the WP to which the specification "belongs".

*** BSW UML Model ***
SWS_CanNm:
———-
Chapter 8.6.1 Optional Interfaces:
Add within SWS_CanNm_00325 the API function Det_ReportRunTimeError

SWS_LinIf:
———-
SWS_LinIf_00359: add Det_ReportRuntimeError

SWS_UdpNm:
———-
Replace UDPNM_E_NO_INIT with UDPNM_E_UNINIT in description of API UdpNm_MainFunction_<Instance Id> (SWS_UdpNm_00234)

*** ECUC XML ***
Not affected. No configuration of runtime error reporting required (see SWS BSW General).
–Last change on issue 59085 comment 88–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.26 Specification Item SWS_TCPIP_00285

Trace References:

SRS_Eth_00129

Content:

If development error detection is enabled: TcpIp_GetAndResetMeasurementData () shall check that the service TcpIp_Init () was previously called. If the check fails, TcpIp_GetAnd ResetMeasurementData () shall raise the development error TCPIP_E_NOTINITUNINIT.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #59085: Rollout of 'Runtime errors'

  Problem description:

  Inconsistencies in SWS with semantics of Default errors
  –Last change on issue 59085 comment 26–

  Agreed solution:

  solution in Column "G" of the new attachment https://www.autosar.org/bugzilla/attachment.cgi?id=4604

  Notes:
  - It is not enough just to migrate the error from one classification table to another. Please also check the related requirements (and background information) which is referring to that error and adapt them if needed.
  - The review task of the ITs shall be done by the WP to which the specification "belongs".

  *** BSW UML Model ***
  SWS_CanNm:
  ———-
  Chapter 8.6.1 Optional Interfaces:
  Add within SWS_CanNm_00325 the API function Det_ReportRunTimeError

  SWS_LinIf:
  ———-
  SWS_LinIf_00359: add Det_ReportRuntimeError

  SWS_UdpNm:
  ———-

Replace UDPNM_E_NO_INIT with UDPNM_E_UNINIT in description of API UdpNm_MainFunction_<Instance Id> (SWS_UdpNm_00234)

*** ECUC XML ***
Not affected. No configuration of runtime error reporting required (see SWS BSW General).
–Last change on issue 59085 comment 88–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.27 Specification Item SWS_TCPIP_00297

Trace References:

SRS_Eth_00016

Content:

If a TcpIpIcmpMsgHandler is configured, the TcpIp shall call the respective <Up>_Icmp MsgHandler() if an ICMPv4 message is received and not handled by the TcpIp directly.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #75094: [TcpIp] Missing behavior description of <Up>_IcmpMsgHandler

  Problem description:

  Currently, there are no requirements for the behavior of the <Up>_IcmpMsgHandler callout function in chapter 7. The function shall be called in case an ICMP error message is received from a remote host and when the message is not handled by the TcpIp e.g. reception of an ICMP Echo Request addressed to a multicast or broadcast address.

  Agreed solution:

  === TcpIp ===
  Add to Chapter 7.2.4 Internet Control Message Protocol (ICMPv4):
  +[SWS_TCPIP_xxxx1] If a TcpIpIcmpMsgHandler is configured, the TcpIp shall call the respective <Up>_IcmpMsgHandler() if an ICMPv4 message is received and not handled by the TcpIp directly.
  Note: For example, if the TcpIp replies to an ICMP echo request <Up>_IcmpMsgHandler() is not called for this message.

Add to 7.3.2 Internet Control Message Protocol (ICMPv6):
+[SWS_TCPIP_xxxx2] If a TcpIpIcmpV6MsgHandler is configured, the TcpIp shall call the respective <Up>_IcmpMsgHandler() if an ICMPv6 message is received and not handled by the TcpIp directly.
Note: For example, if the TcpIp replies to an ICMPv6 echo request <Up>_IcmpMsgHandler() is not called for this message.
–Last change on issue 75094 comment 4–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.28 Specification Item SWS_TCPIP_00298

Trace References:

SRS_Eth_00098

Content:

If a TcpIpIcmpV6MsgHandler is configured, the TcpIp shall call the respective <Up>_Icmp MsgHandler() if an ICMPv6 message is received and not handled by the TcpIp directly.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #75094: [TcpIp] Missing behavior description of <Up>_IcmpMsgHandler

  Problem description:

  Currently, there are no requirements for the behavior of the <Up>_IcmpMsgHandler callout function in chapter 7. The function shall be called in case an ICMP error message is received from a remote host and when the message is not handled by the TcpIp e.g. reception of an ICMP Echo Request addressed to a multicast or broadcast address.

  Agreed solution:

  === TcpIp ===
  Add to Chapter 7.2.4 Internet Control Message Protocol (ICMPv4):
  +[SWS_TCPIP_xxxx1] If a TcpIpIcmpMsgHandler is configured, the TcpIp shall call the respective <Up>_IcmpMsgHandler() if an ICMPv4 message is received and not handled by the TcpIp directly.
  Note: For example, if the TcpIp replies to an ICMP echo request <Up>_IcmpMsgHandler() is not called for this message.

Add to 7.3.2 Internet Control Message Protocol (ICMPv6):
+[SWS_TCPIP_xxxx2] If a TcpIpIcmpV6MsgHandler is configured, the TcpIp shall call the respective <Up>_IcmpMsgHandler() if an ICMPv6 message is received and not handled by the TcpIp directly.
Note: For example, if the TcpIp replies to an ICMPv6 echo request <Up>_IcmpMsgHandler() is not called for this message.
–Last change on issue 75094 comment 4–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.29 Specification Item SWS_TCPIP_00299

Trace References:

none

Content:

In case TcpIp_RequestIpAddrAssignment() is called with parameter Type set to TCPIP_IPADDR_ASSIGNMENT_ALL, the IP address assignment for the IP address table entry specified by LocalAddId shall be initiated for all configured TcpIpAssignment Methods with TcpIpAssignmentTrigger set to TCPIP_MANUAL.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #74847: [DoIP] How to configure DoIPUdpConnection with limited broadcast local IP address

  Problem description:

  according to ISO/FDIS 13400-2:2011 Vehicle Identification Request can be sent to multicast (limited broadcast). However, it is not specified in SWS how such a limited broadcast local address address can be configured for a DoIPUdpConnection.

  Since DoIP shall use SoAd_RequestIpAddrAssignment() with TCPIP_IPADDR_ASSIGNMENT_AUTOIP and TCPIP_IPADDR_ASSIGNMENT_DHCP, it is not possible to use a static local IP address.

  How shall limited broadcast local IP address be configured?
  –Last change on issue 74847 comment 2–

Agreed solution:

SWS DoIP
========

Chapter 7 "Functional Specification"

Change [SWS_DoIP_00204] from:
If the DoIP Activation Line status changes from DOIP_ACTIVATION_LINE_INACTIVE to DOIP_ACTIVATION_LINE_ACTIVE, the DoIP module shall retrieve the SoConId of the first configured UDPConnection, via call to the SoAd_GetSoConId and trigger the IP Address assignment via 2 subsequent calls to SoAd_RequestIpAddrAssignment with the retrieved SoConId, LocalIpAddrPtr set to NULL_PTR and in the first call type set to TCPIP_IPADDR_ASSIGNMENT_LINKLOCAL_DOIP and in the second call type set to TCPIP_IPADDR_ASSIGNMENT_DHCP. (SRS_Eth_00081, SRS_Eth_00028)

To:
If the DoIP Activation Line status changes from DOIP_ACTIVATION_LINE_INACTIVE to DOIP_ACTIVATION_LINE_ACTIVE, the DoIP module shall loop over all DoIPTcpConnection, DoIPUdpConnection, and DoIPUdpVehicleAnnouncementConnections. For each of these DoIPConnections which has a DoIPRequestAddressAssignment set to true the DoIp module shall retrieve the corresponding SoConId via call to the SoAd_GetSoConId and trigger the IP Address assignment via subsequent calls to SoAd_RequestIpAddrAssignment with the retrieved SoConId, LocalIpAddrPtr and DefaultRouterPtr set to NULL_PTR, Netmask set to 0, and Type set to TCPIP_IPADDR_ASSIGNMENT_ALL. For each of these DoIPConnections (irrespective of the value of DoIPRequestAddressAssignment) the DoIP module shall open the respective connection by an according call to SoAd_OpenSoCon. (SRS_Eth_00081, SRS_Eth_00028, SRS_Eth_00026).

Remove the note after SWS_DoIP_00204.

Remove SWS_DoIP_00003

Change [SWS_DoIP_00205] from:
If the function DoIP_SoConModeChg is called with Mode set to SOAD_SOCON_ONLINE, after SWS_DoIP_00003 has been performed, for a UDP vehicle announcement connection, the DoIP module shall send the vehicle announcement message via the corresponding Tx PDU configured in the DoIPUdpVehicleAnnouncementConnection and belonging to the reported socket connection.(SRS_Eth_00026)

To:
If the function DoIP_SoConModeChg is called with Mode set to SOAD_SOCON_ONLINE for a UDP vehicle announcement connection, the DoIP module shall send the vehicle announcement message via the corresponding Tx PDU configured in the DoIPUdpVehicleAnnouncementConnection and belonging to the reported socket connection.(SRS_Eth_00026)

Change [SWS_DoIP_00071] from:
If the DoIP module needs to send a vehicle announcement message because of SWS_DoIP_00003, it shall send the first vehicle announcement message via the configured DoIPUdpVehicleAnnouncementConnection after DoIPInitialVehicleAnnouncementTime as described in Table 6 and repeat this message DoIPVehicleAnnouncementRepetition times with a delay of DoIPVehicleAnnouncementInterval. The last "VIN/GID Status" byte of the Vehicle identification response message is optional as defined in the ISO13400-2 standard. It shall exist only if the "DoIPUseVehicleIdentificationSyncStatus" configuration parameter is set to True. (See SWS_DoIP_00086).(SRS_Eth_00026)

To:
If the DoIP module needs to send a vehicle announcement message (see SWS_DoIP_00205), it shall send the first vehicle announcement message via the configured DoIPUdpVehicleAnnouncementConnection after DoIPInitialVehicleAnnouncementTime as described in Table 6 and repeat this message DoIPVehicleAnnouncementRepetition times with a delay of DoIPVehicleAnnouncementInterval. The last "VIN/GID Status" byte of the Vehicle identification response message is optional as defined in the ISO13400-2 standard. It shall exist only if the "DoIPUseVehicleIdentificationSyncStatus" configuration parameter is set to True. (See SWS_DoIP_00086).(SRS_Eth_00026)

Change [SWS_DoIP_00234] from:
If the DoIP Activation Line status changes from DOIP_ACTIVATION_LINE_ACTIVE to DOIP_ACTIVATION_LINE_INACTIVE, the DoIP module shall retrieve all the SoConId of all the configured UDPConnection, via call to the SoAd_GetSoConId and close all the UDP sockets by calls to the
SoAd_CloseSoCon with the all the retrieved SoConId. (SRS_Eth_00081, SRS_Eth_00028)

To:
If the DoIP Activation Line status changes from DOIP_ACTIVATION_LINE_ACTIVE to DOIP_ACTIVATION_LINE_INACTIVE, the DoIP module shall shall loop over all DoIPTcpConnection, DoIPUdpConnection, and DoIPUdpVehicleAnnounce-

mentConnections. - For each of these DoIPConnections the DoIP module shall retrieve the corresponding SoConId via call to the SoAd_GetSoConId and close all the connection by a call to SoAd_CloseSoCon with the retrieved SoConId. (SRS_Eth_00081, SRS_Eth_00028)

Change [SWS_DoIP_00235] from:
When all UDP sockets are closed (i.e for all the Sockets the function DoIP_SoConModeChg was called with something else than SOAD_SOCON_ONLINE), the DoIP module shall retrieve the SoConId of the first configured UDPConnection, via call to the SoAd_GetSoConId and release the IP
Address assignment via the call to SoAd_ReleaseIpAddrAssignment with the retrieved SoConId. (SRS_Eth_00081, SRS_Eth_00028)

To:
In addition to SWS_DoIP_00234, the DoIP module shall release the corresponding IP Address assignment via the call to SoAd_ReleaseIpAddrAssignment for those connections that have DoIPRequestAddressAssignment set to true. (SRS_Eth_00081, SRS_Eth_00028)

Remove the note after SWS_DoIP_00235.

Chapter 9 "Sequence Diagrams"

Adapt sequence diagrams in section 9.4 "Activation Line Handling Active" and in section 9.5 "Activation Line Handling Inactive" accordingly.

Chapter 10 "Configuration specification"

Add the following new config parameters to DoIPTcpConnection (10.2.13 DoIPTcpConnection), DoIPUdpConnection (10.2.14 DoIPUdpConnection), and DoIPUdpVehicleAnnouncementConnection (10.2.17 DoIPUdpVehicleAnnouncementConnection):

SWS Item ECUC_DoIP_xxxx1 :
Name DoIPRequestAddressAssignment
Description The DoIP module shall request IP address assignment by calling SoAd_RequestIpAddrAssignment() for the TcpIpLocalAddr related to this DoIpConnection.

Multiplicity 1
Type EcucBooleanParamDef
Default value true
Post-Build Variant Value false
Value Configuration Class Pre-compile time X All Variants
Link time –
Post-build time –
Scope / Dependency scope: local

SWS TcpIp
=========

Chapter 8 "API Specification"

Extend [SWS_TCPIP_00065] by a new enumeration literal:
* TCPIP_IPADDR_ASSIGNMENT_ALL - all configured TcpIpAssignmentMethods with TcpIpAssignmentTrigger set to TCPIP_MANUAL

Add an new specification item after [SWS_TCPIP_00080]:
[SWS_TCPIP_xxxx] In case TcpIp_RequestIpAddrAssignment() is called with parameter Type set to TCPIP_IPADDR_ASSIGNMENT_ALL, the IP address assignment for the IP address table entry specified by LocalAddId shall be initiated for all configured TcpIpAssignmentMethods with TcpIpAssignmentTrigger set to TCPIP_MANUAL.
–Last change on issue 74847 comment 42–

BW-C-Level:

Application	Specification	Bus
1	3	1