Document Title	SWS_CANDriver: 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_CANDriver

1.1 Specification Item ECUC_Can_00317

Trace References:

none

Content:

Name	CanRxProcessingCanController.CanRxProcessing		
Parent Container	CanController		
Description	Enables / disables API Can_MainFunction_Read() for handling PDU reception events in polling mode.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	INTERRUPTCan Controller.CanRx Processing.INTERRUPT	Interrupt Mode of operation.	
	MIXEDCanController.CanRx Processing.MIXED	Mixed Mode of operation	
	POLLINGCanController.Can RxProcessing.POLLING	Polling Mode of operation.	
Post-Build Variant Value	false		
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 #66718: [CAN] Can Driver Specification Supports Tx/RxProcessing per Controller not per Mailbox

  Problem description:

  In the Can driver module specification currently TxProcessing/RxProcessing parameters namely CanRxProcessing and CanTxProcessing is configurable per CAN controller but not per MailBox.

  In some of the CAN Hardware there will be interrupts per group of buffers which can be utilized for various use case scenarios.

  Above feature of having configuring those interrupts for a specific group will enable to serve both kind of the messages handling inside a single controller.

A group of buffers interrupt can be disabled when a deterministic timing behavior for a specific group of messages.

But as per the hierarchy of the current AUTOSAR CAN configuration the hardware feature above can't be utilized to the fullest advantage.

Please provide a clarification regards to the same or any changes in the CAN driver configuration will be considered for the above Use case.

Agreed solution:

EcuC
# # # # # # # # # # # # # # # # # # #

In Chapter 10, add to CanHardwareObject:

SWS Item : ECUC_Can_XXXXX :
Name : CanHardwareObjectUsesPolling
Description : Enables polling of this hardware object.
Dependencies : This parameter shall exist if CanRxProcessing/CanTxProcessing is set to Mixed.
Multiplicity : 0..1
Type : Boolean
Default : False

[ECUC_Can_00317]: CanRxProcessing
add new option in Range : Mixed (Mixed Mode of operation)

[ECUC_Can_00318]: CanTxProcessing
add new option in Range : Mixed (Mixed Mode of operation)

Append "or MIXED. In case of MIXED processing only the hardware objects for which CanHardwareObjectUsesPolling is set to TRUE shall be polled." to SWS_Can_00108 and SWS_Can_00031.
–Last change on issue 66718 comment 38–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.2 Specification Item ECUC_Can_00318

Trace References:

none

Content:

Name	CanTxProcessingCanController.CanTxProcessing		
Parent Container	CanController		
Description	Enables / disables API Can_MainFunction_Write() for handling PDU transmission events in polling mode.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	INTERRUPTCan Controller.CanTx Processing.INTERRUPT	Interrupt Mode of operation.	
	MIXEDCanController.CanTx Processing.MIXED	Mixed Mode of operation	
	POLLINGCanController.Can TxProcessing.POLLING	Polling Mode of operation.	
Post-Build Variant Value	false		
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 #66718: [CAN] Can Driver Specification Supports Tx/RxProcessing per Controller not per Mailbox

  Problem description:

  In the Can driver module specification currently TxProcessing/RxProcessing parameters namely CanRxProcessing and CanTxProcessing is configurable per CAN controller but not per MailBox.

  In some of the CAN Hardware there will be interrupts per group of buffers which can be utilized for various use case scenarios.

  Above feature of having configuring those interrupts for a specific group will enable to serve both kind of the messages handling inside a single controller.
  A group of buffers interrupt can be disabled when a deterministic timing behavior for a specific group of messages.

But as per the hierarchy of the current AUTOSAR CAN configuration the hardware feature above can't be utilized to the fullest advantage.

Please provide a clarification regards to the same or any changes in the CAN driver configuration will be considered for the above Use case.

Agreed solution:

EcuC
# # # # # # # # # # # # # # # # # # #

In Chapter 10, add to CanHardwareObject:

SWS Item : ECUC_Can_XXXXX :
Name : CanHardwareObjectUsesPolling
Description : Enables polling of this hardware object.
Dependencies : This parameter shall exist if CanRxProcessing/CanTxProcessing is set to Mixed.
Multiplicity : 0..1
Type : Boolean
Default : False

[ECUC_Can_00317]: CanRxProcessing
add new option in Range : Mixed (Mixed Mode of operation)

[ECUC_Can_00318]: CanTxProcessing
add new option in Range : Mixed (Mixed Mode of operation)

Append "or MIXED. In case of MIXED processing only the hardware objects for which CanHardwareObjectUsesPolling is set to TRUE shall be polled." to SWS_Can_00108 and SWS_Can_00031.
–Last change on issue 66718 comment 38–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.3 Specification Item ECUC_Can_00324

Trace References:

none

Content:

Container Name	CanHardwareObjectCanHardwareObject
Description	This container contains the configuration (parameters) of CAN Hardware Objects.
Configuration Parameters	

Included parameters:

Included Parameters	
Parameter Name	SWS Item ID
CanFdPaddingValue	ECUC_Can_00485
CanHandleType	ECUC_Can_00323
CanHardwareObjectUsesPolling	ECUC_Can_00490
CanHwObjectCount	ECUC_Can_00467
CanIdType	ECUC_Can_00065
CanObjectId	ECUC_Can_00326
CanObjectType	ECUC_Can_00327
CanTriggerTransmitEnable	ECUC_Can_00486
CanControllerRef	ECUC_Can_00322
CanMainFunctionRWPeriodRef	ECUC_Can_00438

Included containers:

Included Containers		
Container Name	Multiplicity	Scope / Dependency
CanHwFilter	0..*	This container is only valid for HRHs and contains the configuration (parameters) of one hardware filter.
CanTTHardwareObjectTrigger	0..*	CanTTHardwareObjectTrigger is specified in the SWS TTCAN and contains the configuration (parameters) of TTCAN triggers for Hardware Objects, which are additional to the configuration (parameters) of CAN Hardware Objects. This container is only included and valid if TTCAN is supported by the controller and, enabled (see CanSupport TTCANRef, ECUC_Can_00430), and used.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #66718: [CAN] Can Driver Specification Supports Tx/RxProcessing per Controller not per Mailbox

  Problem description:

In the Can driver module specification currently TxProcessing/RxProcessing parameters namely CanRxProcessing and CanTxProcessing is configurable per CAN controller but not per MailBox.

In some of the CAN Hardware there will be interrupts per group of buffers which can be utilized for various use case scenarios.

Above feature of having configuring those interrupts for a specific group will enable to serve both kind of the messages handling inside a single controller.
A group of buffers interrupt can be disabled when a deterministic timing behavior for a specific group of messages.

But as per the hierarchy of the current AUTOSAR CAN configuration the hardware feature above can't be utilized to the fullest advantage.

Please provide a clarification regards to the same or any changes in the CAN driver configuration will be considered for the above Use case.

Agreed solution:

EcuC
# # # # # # # # # # # # # # # # # # #

In Chapter 10, add to CanHardwareObject:

SWS Item : ECUC_Can_XXXXX :
Name : CanHardwareObjectUsesPolling
Description : Enables polling of this hardware object.
Dependencies : This parameter shall exist if CanRxProcessing/CanTxProcessing is set to Mixed.
Multiplicity : 0..1
Type : Boolean
Default : False

[ECUC_Can_00317]: CanRxProcessing
add new option in Range : Mixed (Mixed Mode of operation)

[ECUC_Can_00318]: CanTxProcessing
add new option in Range : Mixed (Mixed Mode of operation)

Append "or MIXED. In case of MIXED processing only the hardware objects for which CanHardwareObjectUsesPolling is set to TRUE shall be polled." to

SWS_Can_00108 and SWS_Can_00031.
–Last change on issue 66718 comment 38–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.4 Specification Item ECUC_Can_00490

Trace References:

none

Content:

Name	CanHardwareObjectUsesPollingCanHardwareObject.CanHardwareObjectUsesPolling
Parent Container	CanHardwareObject
Description	Enables polling of this hardware object.
Multiplicity	0..1
Type	EcucBooleanParamDef
Default value	false
Scope / Dependency	dependency: This parameter shall exist if CanRxProcessing/CanTxProcessing is set to Mixed.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #66718: [CAN] Can Driver Specification Supports Tx/RxProcessing per Controller not per Mailbox

  Problem description:

  In the Can driver module specification currently TxProcessing/RxProcessing parameters namely CanRxProcessing and CanTxProcessing is configurable per CAN controller but not per MailBox.

  In some of the CAN Hardware there will be interrupts per group of buffers which can be utilized for various use case scenarios.

  Above feature of having configuring those interrupts for a specific group will enable to serve both kind of the messages handling inside a single controller.
  A group of buffers interrupt can be disabled when a deterministic timing behavior for a specific group of messages.

But as per the hierarchy of the current AUTOSAR CAN configuration the hardware feature above can't be utilized to the fullest advantage.

Please provide a clarification regards to the same or any changes in the CAN driver configuration will be considered for the above Use case.

Agreed solution:

EcuC
# # # # # # # # # # # # # # # # # # #

In Chapter 10, add to CanHardwareObject:

SWS Item : ECUC_Can_XXXXX :
Name : CanHardwareObjectUsesPolling
Description : Enables polling of this hardware object.
Dependencies : This parameter shall exist if CanRxProcessing/CanTxProcessing is set to Mixed.
Multiplicity : 0..1
Type : Boolean
Default : False

[ECUC_Can_00317]: CanRxProcessing
add new option in Range : Mixed (Mixed Mode of operation)

[ECUC_Can_00318]: CanTxProcessing
add new option in Range : Mixed (Mixed Mode of operation)

Append "or MIXED. In case of MIXED processing only the hardware objects for which CanHardwareObjectUsesPolling is set to TRUE shall be polled." to SWS_Can_00108 and SWS_Can_00031.
–Last change on issue 66718 comment 38–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.5 Specification Item SWS_Can_00031

Trace References:

SRS_BSW_00432, SRS_BSW_00373, SRS_SPAL_00157

Content:

The function Can_MainFunction_Write shall perform the polling of TX confirmation when CanTxProcessing

is set to POLLING or MIXED. In case of MIXED processing only the hardware objects for which CanHardwareObjectUsesPolling is set to TRUE shall be polled.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #66718: [CAN] Can Driver Specification Supports Tx/RxProcessing per Controller not per Mailbox

  Problem description:

  In the Can driver module specification currently TxProcessing/RxProcessing parameters namely CanRxProcessing and CanTxProcessing is configurable per CAN controller but not per MailBox.

  In some of the CAN Hardware there will be interrupts per group of buffers which can be utilized for various use case scenarios.

  Above feature of having configuring those interrupts for a specific group will enable to serve both kind of the messages handling inside a single controller.
  A group of buffers interrupt can be disabled when a deterministic timing behavior for a specific group of messages.

  But as per the hierarchy of the current AUTOSAR CAN configuration the hardware feature above can't be utilized to the fullest advantage.

  Please provide a clarification regards to the same or any changes in the CAN driver configuration will be considered for the above Use case.

  Agreed solution:

  EcuC
  # # # # # # # # # # # # # # # # # # #

  In Chapter 10, add to CanHardwareObject:

  SWS Item : ECUC_Can_XXXXX :
  Name : CanHardwareObjectUsesPolling
  Description : Enables polling of this hardware object.
  Dependencies : This parameter shall exist if CanRxProcessing/CanTxProcessing is

set to Mixed.
Multiplicity : 0..1
Type : Boolean
Default : False

[ECUC_Can_00317]: CanRxProcessing
add new option in Range : Mixed (Mixed Mode of operation)

[ECUC_Can_00318]: CanTxProcessing
add new option in Range : Mixed (Mixed Mode of operation)

Append "or MIXED. In case of MIXED processing only the hardware objects for which CanHardwareObjectUsesPolling is set to TRUE shall be polled." to SWS_Can_00108 and SWS_Can_00031.
–Last change on issue 66718 comment 38–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.6 Specification Item SWS_Can_00039

Trace References:

SRS_BSW_00331

Content:

Name:	Can_ReturnTypeCan_ReturnType		
Type:	Enumeration		
Range:	CAN_OKCan_Can_ReturnType.CAN_OK	–	success
	CAN_NOT_OKCan_ReturnType.CAN_NOT_OK	–	error occurred or wakeup event occurred during sleep transition
	CAN_BUSYCan_ReturnType.CAN_BUSY	– 0x02	transmit request could not be processed because no transmit object was available
Description:	Return values of Overlayed return value of Std_ReturnType for CAN driver API . Can_Write()		

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77952: [Can][CanIf] Incompatible return types of Can and CanTrcv

  Problem description:

  While CanTrcv only uses Std_ReturnType, Can uses Can_ReturnType in many places, even when only CAN_OK and CAN_NOT_OK are available.

  This leads to complicated code in CanIf, because it needs to implement separate checks for return values from CanTrcv and Can and cannot just combine the results.

  Agreed solution:

  === CanDrv ===

  Change of SWS_Can_00039 Can_ReturnType:
  * type change from enumeration to extra_litaral
  * Remove range element CAN_OK
  * Remove range element CAN_NOT_OK
  * Assign value "0x02" to range element "CAN_BUSY"
  * Description: Overlayed return value of Std_ReturnType for CAN driver API Can_Write().

  ~SWS_Can_00230 Can_SetControllerMode
  Syntax: Std_ReturnType Can_SetControllerMode(uint8 Controller, Can_StateTransitionType Transition)
  Return value:
  Std_ReturnType
  E_OK: request accepted
  E_NOT_OK: request not accepted, a development error occurred

  ~SWS_Can_00360 Can_CheckWakeup
  Syntax: Std_ReturnType Can_CheckWakeup(uint8 Controller)
  Return value:
  Std_ReturnType
  E_OK: API call has been accepted
  E_NOT_OK: API call has not been accepted

  ~SWS_Can_00233 Can_Write
  Syntax: Std_ReturnType Can_Write(Can_HwHandleType Hth, const Can_PduType* PduInfo)
  Return value:
  Std_ReturnType
  E_OK: Write command has been accepted
  E_NOT_OK: development error occurred

CAN_BUSY: No TX hardware buffer available or pre-emptive call of Can_Write that can't be implemented re-entrant (see Can_ReturnType)

Rename CAN_OK to E_OK and CAN_NOT_OK to E_NOT_OK:
~SWS_Can_00048
~SWS_Can_00089
7.11.5 Return Values
~SWS_Can_00198
~SWS_Can_00199
~SWS_Can_00200
~SWS_Can_00216
~SWS_Can_00217
~SWS_Can_00218
~SWS_CAN_00219
~SWS_CAN_00505
~SWS_CAN_00506
~SWS_Can_00212

=== CanIf ===
Adapt API Can_Write() to new signature:
* Figure 7.10 "Transmission request with multiple CAN Drivers - simplified"
* Figure 9.1 "Transmission request with a single CAN Driver"
* Figure 9.2 "Transmission request with multiple CAN Drivers"
* Figure 9.5 "Transmit confirmation with buffering"
* Figure 9.6 "Transmit Cancelation"
* Figure 9.7 "Trigger Transmit Request"

Adapt API Can_SetControllerMode() to new signature:
* Figure 9.11: Start CAN network
* Figure 9.13: BusOff recovery

Figure 9.13: Change typo "Cnange" to "Change"

~SWS_CANIF_00678: If all calls of Can_CheckWakeup() or CanTrcv_CheckWakeup() return E_NOT_OK to CanIf, then CanIf_CheckWakeup() shall return E_NOT_OK.

~SWS_CANIF_00720: If at least one function call of Can_CheckWakeup() or CanTrcv_CheckWakeup() returns E_OK to CanIf, then CanIf_CheckWakeup() shall return E_OK.

Rename CAN_OK to E_OK and CAN_NOT_OK to E_NOT_OK:
Note between SWS_CANIF_00162 and SWS_CANIF_00319

Table in chapter 9.7 Trigger Transmit Request
Table in chapter 9.11 Start CAN network

=== CanTrcv ===
Adapt API Can_SetControllerMode() to new signature:
* 9.3 De-Initialization (SPI Synchronous)
* 9.4 De-Initialization (SPI Asynchronous)

=== EcuSM ===
Adapt API Can_CheckWakeup() to new signature:
* Figure 42 CAN controller wake up by interrupt
* Figure 43 CAN controller or transceiver wake up by polling

=== TTCanIf ===
Adapt API Can_Write() to new signature:
* Figure 9.1: CAN Interface Time Triggered transmission with Job List

Correct API Can_TTReceive() which has return void instead of Can_ReturnType indeed:
* Figure 9.2: CAN Interface Time Triggered reception with Job List

Rename CAN_OK to E_OK and CAN_NOT_OK to E_NOT_OK:
~SWS_TtCanIf_00071

=== TTCanDrv ===
Rename CAN_OK to E_OK and CAN_NOT_OK to E_NOT_OK:
~SWS_TtCan_00014
~SWS_TtCan_00018
~SWS_TtCan_00022
~SWS_TtCan_00026
~SWS_TtCan_00059
~SWS_TtCan_00078
~SWS_TtCan_00112

=== XCP ===
Adapt API Can_Write() to new signature:
* Figure 5: Xcp on Can Transmit
–Last change on issue 77952 comment 22–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.7 Specification Item SWS_Can_00048

Trace References:

SRS_Can_01122

Content:

In case of a CAN bus wake-up during sleep transition, the function Can_SetController Mode(CAN_CS_STOPPED) shall return CANE_NOT_OK.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77952: [Can][CanIf] Incompatible return types of Can and CanTrcv

  Problem description:

  While CanTrcv only uses Std_ReturnType, Can uses Can_ReturnType in many places, even when only CAN_OK and CAN_NOT_OK are available.

  This leads to complicated code in CanIf, because it needs to implement separate checks for return values from CanTrcv and Can and cannot just combine the results.

  Agreed solution:

  === CanDrv ===

  Change of SWS_Can_00039 Can_ReturnType:
  * type change from enumeration to extra_litaral
  * Remove range element CAN_OK
  * Remove range element CAN_NOT_OK
  * Assign value "0x02" to range element "CAN_BUSY"
  * Description: Overlayed return value of Std_ReturnType for CAN driver API Can_Write().

  ~SWS_Can_00230 Can_SetControllerMode
  Syntax: Std_ReturnType Can_SetControllerMode(uint8 Controller, Can_StateTransitionType Transition)
  Return value:
  Std_ReturnType
  E_OK: request accepted
  E_NOT_OK: request not accepted, a development error occurred

  ~SWS_Can_00360 Can_CheckWakeup
  Syntax: Std_ReturnType Can_CheckWakeup(uint8 Controller)

Return value:
Std_ReturnType
E_OK: API call has been accepted
E_NOT_OK: API call has not been accepted

~SWS_Can_00233 Can_Write
Syntax: Std_ReturnType Can_Write(Can_HwHandleType Hth, const Can_PduType* PduInfo)
Return value:
Std_ReturnType
E_OK: Write command has been accepted
E_NOT_OK: development error occurred
CAN_BUSY: No TX hardware buffer available or pre-emptive call of Can_Write that can't be implemented re-entrant (see Can_ReturnType)

Rename CAN_OK to E_OK and CAN_NOT_OK to E_NOT_OK:
~SWS_Can_00048
~SWS_Can_00089
7.11.5 Return Values
~SWS_Can_00198
~SWS_Can_00199
~SWS_Can_00200
~SWS_Can_00216
~SWS_Can_00217
~SWS_Can_00218
~SWS_CAN_00219
~SWS_CAN_00505
~SWS_CAN_00506
~SWS_Can_00212

=== CanIf ===
Adapt API Can_Write() to new signature:
* Figure 7.10 "Transmission request with multiple CAN Drivers - simplified"
* Figure 9.1 "Transmission request with a single CAN Driver"
* Figure 9.2 "Transmission request with multiple CAN Drivers"
* Figure 9.5 "Transmit confirmation with buffering"
* Figure 9.6 "Transmit Cancelation"
* Figure 9.7 "Trigger Transmit Request"

Adapt API Can_SetControllerMode() to new signature:
* Figure 9.11: Start CAN network
* Figure 9.13: BusOff recovery

Figure 9.13: Change typo "Cnange" to "Change"

~SWS_CANIF_00678: If all calls of Can_CheckWakeup() or CanTrcv_CheckWakeup() return E_NOT_OK to CanIf, then CanIf_CheckWakeup() shall return E_NOT_OK.

~SWS_CANIF_00720: If at least one function call of Can_CheckWakeup() or CanTrcv_CheckWakeup() returns E_OK to CanIf, then
CanIf_CheckWakeup() shall return E_OK.

Rename CAN_OK to E_OK and CAN_NOT_OK to E_NOT_OK:
Note between SWS_CANIF_00162 and SWS_CANIF_00319
Table in chapter 9.7 Trigger Transmit Request
Table in chapter 9.11 Start CAN network

=== CanTrcv ===
Adapt API Can_SetControllerMode() to new signature:
* 9.3 De-Initialization (SPI Synchronous)
* 9.4 De-Initialization (SPI Asynchronous)

=== EcuSM ===
Adapt API Can_CheckWakeup() to new signature:
* Figure 42 CAN controller wake up by interrupt
* Figure 43 CAN controller or transceiver wake up by polling

=== TTCanIf ===
Adapt API Can_Write() to new signature:
* Figure 9.1: CAN Interface Time Triggered transmission with Job List

Correct API Can_TTReceive() which has return void instead of Can_ReturnType indeed:
* Figure 9.2: CAN Interface Time Triggered reception with Job List

Rename CAN_OK to E_OK and CAN_NOT_OK to E_NOT_OK:
~SWS_TtCanIf_00071

=== TTCanDrv ===
Rename CAN_OK to E_OK and CAN_NOT_OK to E_NOT_OK:
~SWS_TtCan_00014
~SWS_TtCan_00018
~SWS_TtCan_00022
~SWS_TtCan_00026
~SWS_TtCan_00059

~SWS_TtCan_00078
~SWS_TtCan_00112

=== XCP ===
Adapt API Can_Write() to new signature:
* Figure 5: Xcp on Can Transmit
–Last change on issue 77952 comment 22–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.8 Specification Item SWS_Can_00089

Trace References:

SRS_BSW_00369, SRS_BSW_00386, SRS_SPAL_12448

Content:

The Can module's environment shall indicate Default development errors only in the return values of a function of the Can module when DET is switched on and the function provides a return value. The returned value is CANE_NOT_OK.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #73570: No "default error" in AUTOSAR

  Problem description:

  The DET was renamed from development error tracer to default error tracer.

  This change was most of the time done automatically and unfortunately renamed "developement error" to "default error".

  "default error" should always be followed by "tracer", otherwise, "development error" is probably the right term.

  This could increase the impact (compared to my selection of impacted document, but formally, the configuration parameters *DevErrorDetect are not using the correct description:
  "Switches the Default Error Tracer (Det) detection and notification..."

  The parameter switches on/off the developement error detection. The DET

does not need to be detected and can be present even when the parameter is set to false.

Agreed solution:

Rename "default error" to "development error" in all impacted documents, but not in an automated way (Do not change "default error tracer" to "developement error tracer"!)

Blueprint/Example:
- sub chapter is now called "7.x Default errors"
- "[SWS_xxx_yyyyy]
In case default error detection is enabled for the xxxx module: The xxxx module shall check API parameters for validity and report detected errors to the DET. ()"
- "[SWS_xxx_yyyyy]
If default error detection is enabled: the function shall check that the service xxx_Init was previously called. If the check fails, the function shall raise the default error XXX_E_NOT_INITIALIZED otherwise (if DET is disabled) return E_NOT_OK. ()"
- "In case default errors are enabled,..."
- "module raises the Default error XXX_E_TRANSITION"
- "The DET provides services to store default errors"
...

The correct text would be:
- sub chapter is called "7.x Development errors"
- "[SWS_xxx_yyyyy]
In case development error detection is enabled for the xxxx module: The xxxx module shall check API parameters for validity and report detected development errors to the DET. ()"
- "[SWS_xxx_yyyyy]
If development error detection is enabled: the function shall check that the service xxx_Init was previously called. If the check fails, the function shall raise the development error XXX_E_NOT_INITIALIZED otherwise (if DET is disabled) return E_NOT_OK. ()"
- "In case development errors are enabled,..."
- "module raises the development error XXX_E_TRANSITION"
- "The DET provides services to store development errors"

Solution for SWS_RTE:
– SWS_RTE —
- Change 4.8 Default errors to 4.8 Development errors
- Change "Errors which can occur at runtime in the RTE are classified as default errors" to "Errors which can occur at runtime in the RTE are classified as development

errors"
- Remove [SWS_Rte_07676]
- Change [SWS_RTE_06611]"If a violation is detected the RTE shall report a default error to the DET." to "If a violation is detected the RTE shall report a development error to the DET."
- Change [SWS_Rte_06631]
[SWS_Rte_06631] d The RTE shall use the OS Application Identifier as the Instance Id to enable the developer to identify in which runtime section of the RTE the error occurs. This Instance ID is even unique across multi cores and so implicitly allows the development error to be traced to a specific core. c(SRS_BSW_00337)

SRS_Libraries:
- In chapter "3 Acronyms and abbreviations": Rename "Development Error Tracer" to "Default Error Tracer"

SRS_SPALGeneral:
- In chapter "6.1.1.3.1 [SRS_SPAL_00157] ...": Rename "Development Error Tracer" to "Default Error Tracer"
- In chapter "6.1.1.4.2 [SRS_SPAL_12448] ...": Rename "Development Error Tracer" to "Default Error Tracer"

SRS_FlashTest:
- In chapter "6.1 Functional Requirements": Rename "Development Error Tracer" to "Default Error Tracer"
- In chapter "7 References":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_MFXLibrary:
- In chapter "2 Acronyms and abbreviations": Rename "Development Error Tracer" to "Default Error Tracer"

SWS_MemoryAbstractionInterface:
- In chapter "3.1 Input documents":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_FlexRayNetworkManagement:
- In chapter "3.3 Related AUTOSAR documents":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_CANStateManager:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_PDURouter:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_EEPROMDriver:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"
–Last change on issue 73570 comment 47–

BW-C-Level:

Application	Specification	Bus
1	1	1

- RfC #77952: [Can][CanIf] Incompatible return types of Can and CanTrcv

  Problem description:

  While CanTrcv only uses Std_ReturnType, Can uses Can_ReturnType in many places, even when only CAN_OK and CAN_NOT_OK are available.

  This leads to complicated code in CanIf, because it needs to implement separate checks for return values from CanTrcv and Can and cannot just combine the results.

  Agreed solution:

  === CanDrv ===

  Change of SWS_Can_00039 Can_ReturnType:
  * type change from enumeration to extra_litaral

* Remove range element CAN_OK
* Remove range element CAN_NOT_OK
* Assign value "0x02" to range element "CAN_BUSY"
* Description: Overlayed return value of Std_ReturnType for CAN driver API Can_Write().

~SWS_Can_00230 Can_SetControllerMode
Syntax: Std_ReturnType Can_SetControllerMode(uint8 Controller, Can_StateTransitionType Transition)
Return value:
Std_ReturnType
E_OK: request accepted
E_NOT_OK: request not accepted, a development error occurred

~SWS_Can_00360 Can_CheckWakeup
Syntax: Std_ReturnType Can_CheckWakeup(uint8 Controller)
Return value:
Std_ReturnType
E_OK: API call has been accepted
E_NOT_OK: API call has not been accepted

~SWS_Can_00233 Can_Write
Syntax: Std_ReturnType Can_Write(Can_HwHandleType Hth, const Can_PduType* PduInfo)
Return value:
Std_ReturnType
E_OK: Write command has been accepted
E_NOT_OK: development error occurred
CAN_BUSY: No TX hardware buffer available or pre-emptive call of Can_Write that can't be implemented re-entrant (see Can_ReturnType)

Rename CAN_OK to E_OK and CAN_NOT_OK to E_NOT_OK:
~SWS_Can_00048
~SWS_Can_00089
7.11.5 Return Values
~SWS_Can_00198
~SWS_Can_00199
~SWS_Can_00200
~SWS_Can_00216
~SWS_Can_00217
~SWS_Can_00218
~SWS_CAN_00219
~SWS_CAN_00505

~SWS_CAN_00506
~SWS_Can_00212

=== CanIf ===
Adapt API Can_Write() to new signature:
* Figure 7.10 "Transmission request with multiple CAN Drivers - simplified"
* Figure 9.1 "Transmission request with a single CAN Driver"
* Figure 9.2 "Transmission request with multiple CAN Drivers"
* Figure 9.5 "Transmit confirmation with buffering"
* Figure 9.6 "Transmit Cancelation"
* Figure 9.7 "Trigger Transmit Request"

Adapt API Can_SetControllerMode() to new signature:
* Figure 9.11: Start CAN network
* Figure 9.13: BusOff recovery

Figure 9.13: Change typo "Cnange" to "Change"

~SWS_CANIF_00678: If all calls of Can_CheckWakeup() or CanTrcv_CheckWakeup() return E_NOT_OK to CanIf, then CanIf_CheckWakeup() shall return E_NOT_OK.

~SWS_CANIF_00720: If at least one function call of Can_CheckWakeup() or CanTrcv_CheckWakeup() returns E_OK to CanIf, then
CanIf_CheckWakeup() shall return E_OK.

Rename CAN_OK to E_OK and CAN_NOT_OK to E_NOT_OK:
Note between SWS_CANIF_00162 and SWS_CANIF_00319
Table in chapter 9.7 Trigger Transmit Request
Table in chapter 9.11 Start CAN network

=== CanTrcv ===
Adapt API Can_SetControllerMode() to new signature:
* 9.3 De-Initialization (SPI Synchronous)
* 9.4 De-Initialization (SPI Asynchronous)

=== EcuSM ===
Adapt API Can_CheckWakeup() to new signature:
* Figure 42 CAN controller wake up by interrupt
* Figure 43 CAN controller or transceiver wake up by polling

=== TTCanIf ===
Adapt API Can_Write() to new signature:

* Figure 9.1: CAN Interface Time Triggered transmission with Job List

Correct API Can_TTReceive() which has return void instead of Can_ReturnType indeed:
* Figure 9.2: CAN Interface Time Triggered reception with Job List

Rename CAN_OK to E_OK and CAN_NOT_OK to E_NOT_OK:
~SWS_TtCanIf_00071

=== TTCanDrv ===
Rename CAN_OK to E_OK and CAN_NOT_OK to E_NOT_OK:
~SWS_TtCan_00014
~SWS_TtCan_00018
~SWS_TtCan_00022
~SWS_TtCan_00026
~SWS_TtCan_00059
~SWS_TtCan_00078
~SWS_TtCan_00112

=== XCP ===
Adapt API Can_Write() to new signature:
* Figure 5: Xcp on Can Transmit
–Last change on issue 77952 comment 22–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.9 Specification Item SWS_Can_00091

Trace References:

SRS_SPAL_12448

Content:

After return of the DET the Can module's function that raised the default development error shall return immediately.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #73570: No "default error" in AUTOSAR

Problem description:

The DET was renamed from development error tracer to default error tracer.

This change was most of the time done automatically and unfortunately renamed "developement error" to "default error".

"default error" should always be followed by "tracer", otherwise, "development error" is probably the right term.

This could increase the impact (compared to my selection of impacted document, but formally, the configuration parameters *DevErrorDetect are not using the correct description:
"Switches the Default Error Tracer (Det) detection and notification..."

The parameter switches on/off the developement error detection. The DET does not need to be detected and can be present even when the parameter is set to false.

Agreed solution:

Rename "default error" to "development error" in all impacted documents, but not in an automated way (Do not change "default error tracer" to "developement error tracer"!)

Blueprint/Example:
- sub chapter is now called "7.x Default errors"
- "[SWS_xxx_yyyyy]
In case default error detection is enabled for the xxxx module: The xxxx module shall check API parameters for validity and report detected errors to the DET. ()"
- "[SWS_xxx_yyyyy]
If default error detection is enabled: the function shall check that the service xxx_Init was previously called. If the check fails, the function shall raise the default error XXX_E_NOT_INITIALIZED otherwise (if DET is disabled) return E_NOT_OK. ()"
- "In case default errors are enabled,..."
- "module raises the Default error XXX_E_TRANSITION"
- "The DET provides services to store default errors"
...

The correct text would be:
- sub chapter is called "7.x Development errors"
- "[SWS_xxx_yyyyy]
In case development error detection is enabled for the xxxx module: The xxxx

module shall check API parameters for validity and report detected development errors to the DET. ()"
- "[SWS_xxx_yyyyy]
If development error detection is enabled: the function shall check that the service xxx_Init was previously called. If the check fails, the function shall raise the development error XXX_E_NOT_INITIALIZED otherwise (if DET is disabled) return E_NOT_OK. ()"
- "In case development errors are enabled,..."
- "module raises the development error XXX_E_TRANSITION"
- "The DET provides services to store development errors"

Solution for SWS_RTE:
– SWS_RTE —
- Change 4.8 Default errors to 4.8 Development errors
- Change "Errors which can occur at runtime in the RTE are classified as default errors" to "Errors which can occur at runtime in the RTE are classified as development errors"
- Remove [SWS_Rte_07676]
- Change [SWS_RTE_06611]"If a violation is detected the RTE shall report a default error to the DET." to "If a violation is detected the RTE shall report a development error to the DET."
- Change [SWS_Rte_06631]
[SWS_Rte_06631] d The RTE shall use the OS Application Identifier as the Instance Id to enable the developer to identify in which runtime section of the RTE the error occurs. This Instance ID is even unique across multi cores and so implicitly allows the development error to be traced to a specific core. c(SRS_BSW_00337)

SRS_Libraries:
- In chapter "3 Acronyms and abbreviations": Rename "Development Error Tracer" to "Default Error Tracer"

SRS_SPALGeneral:
- In chapter "6.1.1.3.1 [SRS_SPAL_00157] ...": Rename "Development Error Tracer" to "Default Error Tracer"
- In chapter "6.1.1.4.2 [SRS_SPAL_12448] ...": Rename "Development Error Tracer" to "Default Error Tracer"

SRS_FlashTest:
- In chapter "6.1 Functional Requirements": Rename "Development Error Tracer" to "Default Error Tracer"

- In chapter "7 References":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_MFXLibrary:
- In chapter "2 Acronyms and abbreviations": Rename "Development Error Tracer" to "Default Error Tracer"

SWS_MemoryAbstractionInterface:
- In chapter "3.1 Input documents":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_FlexRayNetworkManagement:
- In chapter "3.3 Related AUTOSAR documents":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_CANStateManager:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_PDURouter:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_EEPROMDriver:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"
–Last change on issue 73570 comment 47–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.10 Specification Item SWS_Can_00108

Trace References:

SRS_BSW_00432, SRS_SPAL_00157

Content:

The function Can_MainFunction_Read shall perform the polling of RX indications when CanRxProcessing is set to POLLING or MIXED. In case of MIXED processing only the hardware objects for which CanHardwareObjectUsesPolling is set to TRUE shall be polled.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #66718: [CAN] Can Driver Specification Supports Tx/RxProcessing per Controller not per Mailbox

  Problem description:

  In the Can driver module specification currently TxProcessing/RxProcessing parameters namely CanRxProcessing and CanTxProcessing is configurable per CAN controller but not per MailBox.

  In some of the CAN Hardware there will be interrupts per group of buffers which can be utilized for various use case scenarios.

  Above feature of having configuring those interrupts for a specific group will enable to serve both kind of the messages handling inside a single controller.
  A group of buffers interrupt can be disabled when a deterministic timing behavior for a specific group of messages.

  But as per the hierarchy of the current AUTOSAR CAN configuration the hardware feature above can't be utilized to the fullest advantage.

  Please provide a clarification regards to the same or any changes in the CAN driver configuration will be considered for the above Use case.

  Agreed solution:

  EcuC
  # # # # # # # # # # # # # # # # # # #

  In Chapter 10, add to CanHardwareObject:

  SWS Item : ECUC_Can_XXXXX :
  Name : CanHardwareObjectUsesPolling

Description : Enables polling of this hardware object.
Dependencies : This parameter shall exist if CanRxProcessing/CanTxProcessing is set to Mixed.
Multiplicity : 0..1
Type : Boolean
Default : False

[ECUC_Can_00317]: CanRxProcessing
add new option in Range : Mixed (Mixed Mode of operation)

[ECUC_Can_00318]: CanTxProcessing
add new option in Range : Mixed (Mixed Mode of operation)

Append "or MIXED. In case of MIXED processing only the hardware objects for which CanHardwareObjectUsesPolling is set to TRUE shall be polled." to SWS_Can_00108 and SWS_Can_00031.
–Last change on issue 66718 comment 38–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.11 Specification Item SWS_Can_00174

Trace References:

none

Content:

If default development error detection for the Can module is enabled: The function Can_Init shall raise the error CAN_E_TRANSITION if the driver is not in state CAN_UNINIT.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #73570: No "default error" in AUTOSAR

  Problem description:

  The DET was renamed from development error tracer to default error tracer.

  This change was most of the time done automatically and unfortunately re-

named "developement error" to "default error".

"default error" should always be followed by "tracer", otherwise, "development error" is probably the right term.

This could increase the impact (compared to my selection of impacted document, but formally, the configuration parameters *DevErrorDetect are not using the correct description:
"Switches the Default Error Tracer (Det) detection and notification..."

The parameter switches on/off the developement error detection. The DET does not need to be detected and can be present even when the parameter is set to false.

Agreed solution:

Rename "default error" to "development error" in all impacted documents, but not in an automated way (Do not change "default error tracer" to "developement error tracer"!)

Blueprint/Example:
- sub chapter is now called "7.x Default errors"
- "[SWS_xxx_yyyyy]
In case default error detection is enabled for the xxxx module: The xxxx module shall check API parameters for validity and report detected errors to the DET. ()"
- "[SWS_xxx_yyyyy]
If default error detection is enabled: the function shall check that the service xxx_Init was previously called. If the check fails, the function shall raise the default error XXX_E_NOT_INITIALIZED otherwise (if DET is disabled) return E_NOT_OK. ()"
- "In case default errors are enabled,..."
- "module raises the Default error XXX_E_TRANSITION"
- "The DET provides services to store default errors"
...

The correct text would be:
- sub chapter is called "7.x Development errors"
- "[SWS_xxx_yyyyy]
In case development error detection is enabled for the xxxx module: The xxxx module shall check API parameters for validity and report detected development errors to the DET. ()"
- "[SWS_xxx_yyyyy]
If development error detection is enabled: the function shall check that the service xxx_Init was previously called. If the check fails, the function shall raise the

development error XXX_E_NOT_INITIALIZED otherwise (if DET is disabled) return E_NOT_OK. ()"
- "In case development errors are enabled,..."
- "module raises the development error XXX_E_TRANSITION"
- "The DET provides services to store development errors"

Solution for SWS_RTE:
– SWS_RTE —
- Change 4.8 Default errors to 4.8 Development errors
- Change "Errors which can occur at runtime in the RTE are classified as default errors" to "Errors which can occur at runtime in the RTE are classified as development errors"
- Remove [SWS_Rte_07676]
- Change [SWS_RTE_06611]"If a violation is detected the RTE shall report a default error to the DET." to "If a violation is detected the RTE shall report a development error to the DET."
- Change [SWS_Rte_06631]
[SWS_Rte_06631] d The RTE shall use the OS Application Identifier as the Instance Id to enable the developer to identify in which runtime section of the RTE the error occurs. This Instance ID is even unique across multi cores and so implicitly allows the development error to be traced to a specific core. c(SRS_BSW_00337)

SRS_Libraries:
- In chapter "3 Acronyms and abbreviations": Rename "Development Error Tracer" to "Default Error Tracer"

SRS_SPALGeneral:
- In chapter "6.1.1.3.1 [SRS_SPAL_00157] ...": Rename "Development Error Tracer" to "Default Error Tracer"
- In chapter "6.1.1.4.2 [SRS_SPAL_12448] ...": Rename "Development Error Tracer" to "Default Error Tracer"

SRS_FlashTest:
- In chapter "6.1 Functional Requirements": Rename "Development Error Tracer" to "Default Error Tracer"
- In chapter "7 References":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_MFXLibrary:
- In chapter "2 Acronyms and abbreviations": Rename "Development Error Tracer" to "Default Error Tracer"

SWS_MemoryAbstractionInterface:
- In chapter "3.1 Input documents":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_FlexRayNetworkManagement:
- In chapter "3.3 Related AUTOSAR documents":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_CANStateManager:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_PDURouter:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_EEPROMDriver:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"
–Last change on issue 73570 comment 47–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.12 Specification Item SWS_Can_00177

Trace References:

none

Content:

If default development error detection for the Can module is enabled: The function Can_GetVersionInfo shall raise the error CAN_E_PARAM_POINTER if the parameter versionInfo is a null pointer.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #73570: No "default error" in AUTOSAR

  Problem description:

  The DET was renamed from development error tracer to default error tracer.

  This change was most of the time done automatically and unfortunately renamed "developement error" to "default error".

  "default error" should always be followed by "tracer", otherwise, "development error" is probably the right term.

  This could increase the impact (compared to my selection of impacted document, but formally, the configuration parameters *DevErrorDetect are not using the correct description:
  "Switches the Default Error Tracer (Det) detection and notification..."

  The parameter switches on/off the developement error detection. The DET does not need to be detected and can be present even when the parameter is set to false.

  Agreed solution:

  Rename "default error" to "development error" in all impacted documents, but not in an automated way (Do not change "default error tracer" to "developement error tracer"!)

  Blueprint/Example:
  - sub chapter is now called "7.x Default errors"
  - "[SWS_xxx_yyyyy]
  In case default error detection is enabled for the xxxx module: The xxxx module shall check API parameters for validity and report detected errors to the DET. ()"

- "[SWS_xxx_yyyyy]
If default error detection is enabled: the function shall check that the service xxx_Init was previously called. If the check fails, the function shall raise the default error XXX_E_NOT_INITIALIZED otherwise (if DET is disabled) return E_NOT_OK. ()"
- "In case default errors are enabled,..."
- "module raises the Default error XXX_E_TRANSITION"
- "The DET provides services to store default errors"
...

The correct text would be:
- sub chapter is called "7.x Development errors"
- "[SWS_xxx_yyyyy]
In case development error detection is enabled for the xxxx module: The xxxx module shall check API parameters for validity and report detected development errors to the DET. ()"
- "[SWS_xxx_yyyyy]
If development error detection is enabled: the function shall check that the service xxx_Init was previously called. If the check fails, the function shall raise the development error XXX_E_NOT_INITIALIZED otherwise (if DET is disabled) return E_NOT_OK. ()"
- "In case development errors are enabled,..."
- "module raises the development error XXX_E_TRANSITION"
- "The DET provides services to store development errors"

Solution for SWS_RTE:
– SWS_RTE —
- Change 4.8 Default errors to 4.8 Development errors
- Change "Errors which can occur at runtime in the RTE are classified as default errors" to "Errors which can occur at runtime in the RTE are classified as development errors"
- Remove [SWS_Rte_07676]
- Change [SWS_RTE_06611]"If a violation is detected the RTE shall report a default error to the DET." to "If a violation is detected the RTE shall report a development error to the DET."
- Change [SWS_Rte_06631]
[SWS_Rte_06631] d The RTE shall use the OS Application Identifier as the Instance Id to enable the developer to identify in which runtime section of the RTE the error occurs. This Instance ID is even unique across multi cores and so implicitly allows the development error to be traced to a specific core. c(SRS_BSW_00337)

SRS_Libraries:

- In chapter "3 Acronyms and abbreviations": Rename "Development Error Tracer" to "Default Error Tracer"

SRS_SPALGeneral:
- In chapter "6.1.1.3.1 [SRS_SPAL_00157] ...": Rename "Development Error Tracer" to "Default Error Tracer"
- In chapter "6.1.1.4.2 [SRS_SPAL_12448] ...": Rename "Development Error Tracer" to "Default Error Tracer"

SRS_FlashTest:
- In chapter "6.1 Functional Requirements": Rename "Development Error Tracer" to "Default Error Tracer"
- In chapter "7 References":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_MFXLibrary:
- In chapter "2 Acronyms and abbreviations": Rename "Development Error Tracer" to "Default Error Tracer"

SWS_MemoryAbstractionInterface:
- In chapter "3.1 Input documents":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_FlexRayNetworkManagement:
- In chapter "3.3 Related AUTOSAR documents":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_CANStateManager:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_PDURouter:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_EEPROMDriver:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"
–Last change on issue 73570 comment 47–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.13 Specification Item SWS_Can_00198

Trace References:

none

Content:

If default development error detection for the Can module is enabled: if the module is not yet initialized, the function Can_SetControllerMode shall raise default development error CAN_E_UNINIT and return CANE_NOT_OK.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #73570: No "default error" in AUTOSAR

  Problem description:

  The DET was renamed from development error tracer to default error tracer.

  This change was most of the time done automatically and unfortunately renamed "developement error" to "default error".

  "default error" should always be followed by "tracer", otherwise, "development error" is probably the right term.

  This could increase the impact (compared to my selection of impacted document, but formally, the configuration parameters *DevErrorDetect are not using the correct description:

"Switches the Default Error Tracer (Det) detection and notification..."

The parameter switches on/off the developement error detection. The DET does not need to be detected and can be present even when the parameter is set to false.

Agreed solution:

Rename "default error" to "development error" in all impacted documents, but not in an automated way (Do not change "default error tracer" to "developement error tracer"!)

Blueprint/Example:
- sub chapter is now called "7.x Default errors"
- "[SWS_xxx_yyyyy]
In case default error detection is enabled for the xxxx module: The xxxx module shall check API parameters for validity and report detected errors to the DET. ()"
- "[SWS_xxx_yyyyy]
If default error detection is enabled: the function shall check that the service xxx_Init was previously called. If the check fails, the function shall raise the default error XXX_E_NOT_INITIALIZED otherwise (if DET is disabled) return E_NOT_OK. ()"
- "In case default errors are enabled,..."
- "module raises the Default error XXX_E_TRANSITION"
- "The DET provides services to store default errors"
...

The correct text would be:
- sub chapter is called "7.x Development errors"
- "[SWS_xxx_yyyyy]
In case development error detection is enabled for the xxxx module: The xxxx module shall check API parameters for validity and report detected development errors to the DET. ()"
- "[SWS_xxx_yyyyy]
If development error detection is enabled: the function shall check that the service xxx_Init was previously called. If the check fails, the function shall raise the development error XXX_E_NOT_INITIALIZED otherwise (if DET is disabled) return E_NOT_OK. ()"
- "In case development errors are enabled,..."
- "module raises the development error XXX_E_TRANSITION"
- "The DET provides services to store development errors"

Solution for SWS_RTE:
– SWS_RTE —

- Change 4.8 Default errors to 4.8 Development errors
- Change "Errors which can occur at runtime in the RTE are classified as default errors" to "Errors which can occur at runtime in the RTE are classified as development errors"
- Remove [SWS_Rte_07676]
- Change [SWS_RTE_06611]"If a violation is detected the RTE shall report a default error to the DET." to "If a violation is detected the RTE shall report a development error to the DET."
- Change [SWS_Rte_06631]
[SWS_Rte_06631] d The RTE shall use the OS Application Identifier as the Instance Id to enable the developer to identify in which runtime section of the RTE the error occurs. This Instance ID is even unique across multi cores and so implicitly allows the development error to be traced to a specific core. c(SRS_BSW_00337)

SRS_Libraries:
- In chapter "3 Acronyms and abbreviations": Rename "Development Error Tracer" to "Default Error Tracer"

SRS_SPALGeneral:
- In chapter "6.1.1.3.1 [SRS_SPAL_00157] ...": Rename "Development Error Tracer" to "Default Error Tracer"
- In chapter "6.1.1.4.2 [SRS_SPAL_12448] ...": Rename "Development Error Tracer" to "Default Error Tracer"

SRS_FlashTest:
- In chapter "6.1 Functional Requirements": Rename "Development Error Tracer" to "Default Error Tracer"
- In chapter "7 References":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_MFXLibrary:
- In chapter "2 Acronyms and abbreviations": Rename "Development Error Tracer" to "Default Error Tracer"

SWS_MemoryAbstractionInterface:
- In chapter "3.1 Input documents":

Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_FlexRayNetworkManagement:
- In chapter "3.3 Related AUTOSAR documents":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_CANStateManager:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_PDURouter:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_EEPROMDriver:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"
–Last change on issue 73570 comment 47–

BW-C-Level:

Application	Specification	Bus
1	1	1

- RfC #77952: [Can][CanIf] Incompatible return types of Can and CanTrcv

  Problem description:

  While CanTrcv only uses Std_ReturnType, Can uses Can_ReturnType in many places, even when only CAN_OK and CAN_NOT_OK are available.

  This leads to complicated code in CanIf, because it needs to implement separate checks for return values from CanTrcv and Can and cannot just combine the results.

  Agreed solution:

=== CanDrv ===

Change of SWS_Can_00039 Can_ReturnType:
* type change from enumeration to extra_litaral
* Remove range element CAN_OK
* Remove range element CAN_NOT_OK
* Assign value "0x02" to range element "CAN_BUSY"
* Description: Overlayed return value of Std_ReturnType for CAN driver API Can_Write().

~SWS_Can_00230 Can_SetControllerMode
Syntax: Std_ReturnType Can_SetControllerMode(uint8 Controller, Can_StateTransitionType Transition)
Return value:
Std_ReturnType
E_OK: request accepted
E_NOT_OK: request not accepted, a development error occurred

~SWS_Can_00360 Can_CheckWakeup
Syntax: Std_ReturnType Can_CheckWakeup(uint8 Controller)
Return value:
Std_ReturnType
E_OK: API call has been accepted
E_NOT_OK: API call has not been accepted

~SWS_Can_00233 Can_Write
Syntax: Std_ReturnType Can_Write(Can_HwHandleType Hth, const Can_PduType* PduInfo)
Return value:
Std_ReturnType
E_OK: Write command has been accepted
E_NOT_OK: development error occurred
CAN_BUSY: No TX hardware buffer available or pre-emptive call of Can_Write that can't be implemented re-entrant (see Can_ReturnType)

Rename CAN_OK to E_OK and CAN_NOT_OK to E_NOT_OK:
~SWS_Can_00048
~SWS_Can_00089
7.11.5 Return Values
~SWS_Can_00198
~SWS_Can_00199
~SWS_Can_00200
~SWS_Can_00216

~SWS_Can_00217
~SWS_Can_00218
~SWS_CAN_00219
~SWS_CAN_00505
~SWS_CAN_00506
~SWS_Can_00212

=== CanIf ===
Adapt API Can_Write() to new signature:
* Figure 7.10 "Transmission request with multiple CAN Drivers - simplified"
* Figure 9.1 "Transmission request with a single CAN Driver"
* Figure 9.2 "Transmission request with multiple CAN Drivers"
* Figure 9.5 "Transmit confirmation with buffering"
* Figure 9.6 "Transmit Cancelation"
* Figure 9.7 "Trigger Transmit Request"

Adapt API Can_SetControllerMode() to new signature:
* Figure 9.11: Start CAN network
* Figure 9.13: BusOff recovery

Figure 9.13: Change typo "Cnange" to "Change"

~SWS_CANIF_00678: If all calls of Can_CheckWakeup() or CanTrcv_CheckWakeup() return E_NOT_OK to CanIf, then CanIf_CheckWakeup() shall return E_NOT_OK.

~SWS_CANIF_00720: If at least one function call of Can_CheckWakeup() or CanTrcv_CheckWakeup() returns E_OK to CanIf, then
CanIf_CheckWakeup() shall return E_OK.

Rename CAN_OK to E_OK and CAN_NOT_OK to E_NOT_OK:
Note between SWS_CANIF_00162 and SWS_CANIF_00319
Table in chapter 9.7 Trigger Transmit Request
Table in chapter 9.11 Start CAN network

=== CanTrcv ===
Adapt API Can_SetControllerMode() to new signature:
* 9.3 De-Initialization (SPI Synchronous)
* 9.4 De-Initialization (SPI Asynchronous)

=== EcuSM ===
Adapt API Can_CheckWakeup() to new signature:
* Figure 42 CAN controller wake up by interrupt

* Figure 43 CAN controller or transceiver wake up by polling

=== TTCanIf ===
Adapt API Can_Write() to new signature:
* Figure 9.1: CAN Interface Time Triggered transmission with Job List

Correct API Can_TTReceive() which has return void instead of Can_ReturnType indeed:
* Figure 9.2: CAN Interface Time Triggered reception with Job List

Rename CAN_OK to E_OK and CAN_NOT_OK to E_NOT_OK:
~SWS_TtCanIf_00071

=== TTCanDrv ===
Rename CAN_OK to E_OK and CAN_NOT_OK to E_NOT_OK:
~SWS_TtCan_00014
~SWS_TtCan_00018
~SWS_TtCan_00022
~SWS_TtCan_00026
~SWS_TtCan_00059
~SWS_TtCan_00078
~SWS_TtCan_00112

=== XCP ===
Adapt API Can_Write() to new signature:
* Figure 5: Xcp on Can Transmit
–Last change on issue 77952 comment 22–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.14 Specification Item SWS_Can_00199

Trace References:

none

Content:

If default development error detection for the Can module is enabled: if the parameter Controller is out of range, the function Can_SetControllerMode shall raise default development error CAN_E_PARAM_CONTROLLER and return CANE_NOT_OK.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #73570: No "default error" in AUTOSAR

  Problem description:

  The DET was renamed from development error tracer to default error tracer.

  This change was most of the time done automatically and unfortunately renamed "developement error" to "default error".

  "default error" should always be followed by "tracer", otherwise, "development error" is probably the right term.

  This could increase the impact (compared to my selection of impacted document, but formally, the configuration parameters *DevErrorDetect are not using the correct description:
  "Switches the Default Error Tracer (Det) detection and notification..."

  The parameter switches on/off the developement error detection. The DET does not need to be detected and can be present even when the parameter is set to false.

  Agreed solution:

  Rename "default error" to "development error" in all impacted documents, but not in an automated way (Do not change "default error tracer" to "developement error tracer"!)

  Blueprint/Example:
  - sub chapter is now called "7.x Default errors"
  - "[SWS_xxx_yyyyy]
  In case default error detection is enabled for the xxxx module: The xxxx module shall check API parameters for validity and report detected errors to the DET. ()"
  - "[SWS_xxx_yyyyy]
  If default error detection is enabled: the function shall check that the service xxx_Init was previously called. If the check fails, the function shall raise the default error XXX_E_NOT_INITIALIZED otherwise (if DET is disabled) return E_NOT_OK. ()"
  - "In case default errors are enabled,..."
  - "module raises the Default error XXX_E_TRANSITION"
  - "The DET provides services to store default errors"
  ...

  The correct text would be:

- sub chapter is called "7.x Development errors"
- "[SWS_xxx_yyyyy]
In case development error detection is enabled for the xxxx module: The xxxx module shall check API parameters for validity and report detected development errors to the DET. ()"
- "[SWS_xxx_yyyyy]
If development error detection is enabled: the function shall check that the service xxx_Init was previously called. If the check fails, the function shall raise the development error XXX_E_NOT_INITIALIZED otherwise (if DET is disabled) return E_NOT_OK. ()"
- "In case development errors are enabled,..."
- "module raises the development error XXX_E_TRANSITION"
- "The DET provides services to store development errors"

Solution for SWS_RTE:
– SWS_RTE —
- Change 4.8 Default errors to 4.8 Development errors
- Change "Errors which can occur at runtime in the RTE are classified as default errors" to "Errors which can occur at runtime in the RTE are classified as development errors"
- Remove [SWS_Rte_07676]
- Change [SWS_RTE_06611]"If a violation is detected the RTE shall report a default error to the DET." to "If a violation is detected the RTE shall report a development error to the DET."
- Change [SWS_Rte_06631]
[SWS_Rte_06631] d The RTE shall use the OS Application Identifier as the Instance Id to enable the developer to identify in which runtime section of the RTE the error occurs. This Instance ID is even unique across multi cores and so implicitly allows the development error to be traced to a specific core. c(SRS_BSW_00337)

SRS_Libraries:
- In chapter "3 Acronyms and abbreviations": Rename "Development Error Tracer" to "Default Error Tracer"

SRS_SPALGeneral:
- In chapter "6.1.1.3.1 [SRS_SPAL_00157] ...": Rename "Development Error Tracer" to "Default Error Tracer"
- In chapter "6.1.1.4.2 [SRS_SPAL_12448] ...": Rename "Development Error Tracer" to "Default Error Tracer"

SRS_FlashTest:
- In chapter "6.1 Functional Requirements": Rename "Development Error Tracer" to "Default Error Tracer"
- In chapter "7 References":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_MFXLibrary:
- In chapter "2 Acronyms and abbreviations": Rename "Development Error Tracer" to "Default Error Tracer"

SWS_MemoryAbstractionInterface:
- In chapter "3.1 Input documents":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_FlexRayNetworkManagement:
- In chapter "3.3 Related AUTOSAR documents":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_CANStateManager:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_PDURouter:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_EEPROMDriver:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"
–Last change on issue 73570 comment 47–

BW-C-Level:

Application	Specification	Bus
1	1	1

- RfC #77952: [Can][CanIf] Incompatible return types of Can and CanTrcv

  Problem description:

  While CanTrcv only uses Std_ReturnType, Can uses Can_ReturnType in many places, even when only CAN_OK and CAN_NOT_OK are available.

  This leads to complicated code in CanIf, because it needs to implement separate checks for return values from CanTrcv and Can and cannot just combine the results.

  Agreed solution:

  === CanDrv ===

  Change of SWS_Can_00039 Can_ReturnType:
  * type change from enumeration to extra_litaral
  * Remove range element CAN_OK
  * Remove range element CAN_NOT_OK
  * Assign value "0x02" to range element "CAN_BUSY"
  * Description: Overlayed return value of Std_ReturnType for CAN driver API Can_Write().

  ~SWS_Can_00230 Can_SetControllerMode
  Syntax: Std_ReturnType Can_SetControllerMode(uint8 Controller, Can_StateTransitionType Transition)
  Return value:
  Std_ReturnType
  E_OK: request accepted
  E_NOT_OK: request not accepted, a development error occurred

  ~SWS_Can_00360 Can_CheckWakeup
  Syntax: Std_ReturnType Can_CheckWakeup(uint8 Controller)
  Return value:
  Std_ReturnType
  E_OK: API call has been accepted
  E_NOT_OK: API call has not been accepted

  ~SWS_Can_00233 Can_Write
  Syntax: Std_ReturnType Can_Write(Can_HwHandleType Hth, const Can_PduType* PduInfo)

Return value:
Std_ReturnType
E_OK: Write command has been accepted
E_NOT_OK: development error occurred
CAN_BUSY: No TX hardware buffer available or pre-emptive call of Can_Write that can't be implemented re-entrant (see Can_ReturnType)

Rename CAN_OK to E_OK and CAN_NOT_OK to E_NOT_OK:
~SWS_Can_00048
~SWS_Can_00089
7.11.5 Return Values
~SWS_Can_00198
~SWS_Can_00199
~SWS_Can_00200
~SWS_Can_00216
~SWS_Can_00217
~SWS_Can_00218
~SWS_CAN_00219
~SWS_CAN_00505
~SWS_CAN_00506
~SWS_Can_00212

=== CanIf ===
Adapt API Can_Write() to new signature:
* Figure 7.10 "Transmission request with multiple CAN Drivers - simplified"
* Figure 9.1 "Transmission request with a single CAN Driver"
* Figure 9.2 "Transmission request with multiple CAN Drivers"
* Figure 9.5 "Transmit confirmation with buffering"
* Figure 9.6 "Transmit Cancelation"
* Figure 9.7 "Trigger Transmit Request"

Adapt API Can_SetControllerMode() to new signature:
* Figure 9.11: Start CAN network
* Figure 9.13: BusOff recovery

Figure 9.13: Change typo "Cnange" to "Change"

~SWS_CANIF_00678: If all calls of Can_CheckWakeup() or CanTrcv_CheckWakeup() return E_NOT_OK to CanIf, then CanIf_CheckWakeup() shall return E_NOT_OK.

~SWS_CANIF_00720: If at least one function call of Can_CheckWakeup() or CanTrcv_CheckWakeup() returns E_OK to CanIf, then

CanIf_CheckWakeup() shall return E_OK.

Rename CAN_OK to E_OK and CAN_NOT_OK to E_NOT_OK:
Note between SWS_CANIF_00162 and SWS_CANIF_00319
Table in chapter 9.7 Trigger Transmit Request
Table in chapter 9.11 Start CAN network

=== CanTrcv ===
Adapt API Can_SetControllerMode() to new signature:
* 9.3 De-Initialization (SPI Synchronous)
* 9.4 De-Initialization (SPI Asynchronous)

=== EcuSM ===
Adapt API Can_CheckWakeup() to new signature:
* Figure 42 CAN controller wake up by interrupt
* Figure 43 CAN controller or transceiver wake up by polling

=== TTCanIf ===
Adapt API Can_Write() to new signature:
* Figure 9.1: CAN Interface Time Triggered transmission with Job List

Correct API Can_TTReceive() which has return void instead of Can_ReturnType indeed:
* Figure 9.2: CAN Interface Time Triggered reception with Job List

Rename CAN_OK to E_OK and CAN_NOT_OK to E_NOT_OK:
~SWS_TtCanIf_00071

=== TTCanDrv ===
Rename CAN_OK to E_OK and CAN_NOT_OK to E_NOT_OK:
~SWS_TtCan_00014
~SWS_TtCan_00018
~SWS_TtCan_00022
~SWS_TtCan_00026
~SWS_TtCan_00059
~SWS_TtCan_00078
~SWS_TtCan_00112

=== XCP ===
Adapt API Can_Write() to new signature:
* Figure 5: Xcp on Can Transmit
–Last change on issue 77952 comment 22–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.15 Specification Item SWS_Can_00200

Trace References:

none

Content:

If default development error detection for the Can module is enabled: if an invalid transition has been requested, the function Can_SetControllerMode shall raise the error CAN_E_TRANSITION and return CANE_NOT_OK.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #73570: No "default error" in AUTOSAR

  Problem description:

  The DET was renamed from development error tracer to default error tracer.

  This change was most of the time done automatically and unfortunately renamed "developement error" to "default error".

  "default error" should always be followed by "tracer", otherwise, "development error" is probably the right term.

  This could increase the impact (compared to my selection of impacted document, but formally, the configuration parameters *DevErrorDetect are not using the correct description:
  "Switches the Default Error Tracer (Det) detection and notification..."

  The parameter switches on/off the developement error detection. The DET does not need to be detected and can be present even when the parameter is set to false.

  Agreed solution:

  Rename "default error" to "development error" in all impacted documents, but not in an automated way (Do not change "default error tracer" to "developement error tracer"!)

Blueprint/Example:
- sub chapter is now called "7.x Default errors"
- "[SWS_xxx_yyyyy]
In case default error detection is enabled for the xxxx module: The xxxx module shall check API parameters for validity and report detected errors to the DET. ()"
- "[SWS_xxx_yyyyy]
If default error detection is enabled: the function shall check that the service xxx_Init was previously called. If the check fails, the function shall raise the default error XXX_E_NOT_INITIALIZED otherwise (if DET is disabled) return E_NOT_OK. ()"
- "In case default errors are enabled,..."
- "module raises the Default error XXX_E_TRANSITION"
- "The DET provides services to store default errors"
...

The correct text would be:
- sub chapter is called "7.x Development errors"
- "[SWS_xxx_yyyyy]
In case development error detection is enabled for the xxxx module: The xxxx module shall check API parameters for validity and report detected development errors to the DET. ()"
- "[SWS_xxx_yyyyy]
If development error detection is enabled: the function shall check that the service xxx_Init was previously called. If the check fails, the function shall raise the development error XXX_E_NOT_INITIALIZED otherwise (if DET is disabled) return E_NOT_OK. ()"
- "In case development errors are enabled,..."
- "module raises the development error XXX_E_TRANSITION"
- "The DET provides services to store development errors"

Solution for SWS_RTE:
– SWS_RTE —
- Change 4.8 Default errors to 4.8 Development errors
- Change "Errors which can occur at runtime in the RTE are classified as default errors" to "Errors which can occur at runtime in the RTE are classified as development errors"
- Remove [SWS_Rte_07676]
- Change [SWS_RTE_06611]"If a violation is detected the RTE shall report a default error to the DET." to "If a violation is detected the RTE shall report a development error to the DET."
- Change [SWS_Rte_06631]
[SWS_Rte_06631] d The RTE shall use the OS Application Identifier as the Instance

Id to enable the developer to identify in which runtime section of the RTE the error occurs. This Instance ID is even unique across multi cores and so implicitly allows the development error to be traced to a specific core. c(SRS_BSW_00337)

SRS_Libraries:
- In chapter "3 Acronyms and abbreviations": Rename "Development Error Tracer" to "Default Error Tracer"

SRS_SPALGeneral:
- In chapter "6.1.1.3.1 [SRS_SPAL_00157] ...": Rename "Development Error Tracer" to "Default Error Tracer"
- In chapter "6.1.1.4.2 [SRS_SPAL_12448] ...": Rename "Development Error Tracer" to "Default Error Tracer"

SRS_FlashTest:
- In chapter "6.1 Functional Requirements": Rename "Development Error Tracer" to "Default Error Tracer"
- In chapter "7 References":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_MFXLibrary:
- In chapter "2 Acronyms and abbreviations": Rename "Development Error Tracer" to "Default Error Tracer"

SWS_MemoryAbstractionInterface:
- In chapter "3.1 Input documents":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_FlexRayNetworkManagement:
- In chapter "3.3 Related AUTOSAR documents":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_CANStateManager:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_PDURouter:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_EEPROMDriver:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"
–Last change on issue 73570 comment 47–

BW-C-Level:

Application	Specification	Bus
1	1	1

- RfC #77952: [Can][CanIf] Incompatible return types of Can and CanTrcv

  Problem description:

  While CanTrcv only uses Std_ReturnType, Can uses Can_ReturnType in many places, even when only CAN_OK and CAN_NOT_OK are available.

  This leads to complicated code in CanIf, because it needs to implement separate checks for return values from CanTrcv and Can and cannot just combine the results.

  Agreed solution:

  === CanDrv ===

  Change of SWS_Can_00039 Can_ReturnType:
  * type change from enumeration to extra_litaral
  * Remove range element CAN_OK
  * Remove range element CAN_NOT_OK
  * Assign value "0x02" to range element "CAN_BUSY"
  * Description: Overlayed return value of Std_ReturnType for CAN driver API Can_Write().

  ~SWS_Can_00230 Can_SetControllerMode

Syntax: Std_ReturnType Can_SetControllerMode(uint8 Controller, Can_StateTransitionType Transition)
Return value:
Std_ReturnType
E_OK: request accepted
E_NOT_OK: request not accepted, a development error occurred

~SWS_Can_00360 Can_CheckWakeup
Syntax: Std_ReturnType Can_CheckWakeup(uint8 Controller)
Return value:
Std_ReturnType
E_OK: API call has been accepted
E_NOT_OK: API call has not been accepted

~SWS_Can_00233 Can_Write
Syntax: Std_ReturnType Can_Write(Can_HwHandleType Hth, const Can_PduType* PduInfo)
Return value:
Std_ReturnType
E_OK: Write command has been accepted
E_NOT_OK: development error occurred
CAN_BUSY: No TX hardware buffer available or pre-emptive call of Can_Write that can't be implemented re-entrant (see Can_ReturnType)

Rename CAN_OK to E_OK and CAN_NOT_OK to E_NOT_OK:
~SWS_Can_00048
~SWS_Can_00089
7.11.5 Return Values
~SWS_Can_00198
~SWS_Can_00199
~SWS_Can_00200
~SWS_Can_00216
~SWS_Can_00217
~SWS_Can_00218
~SWS_CAN_00219
~SWS_CAN_00505
~SWS_CAN_00506
~SWS_Can_00212

=== CanIf ===
Adapt API Can_Write() to new signature:
* Figure 7.10 "Transmission request with multiple CAN Drivers - simplified"
* Figure 9.1 "Transmission request with a single CAN Driver"

* Figure 9.2 "Transmission request with multiple CAN Drivers"
* Figure 9.5 "Transmit confirmation with buffering"
* Figure 9.6 "Transmit Cancelation"
* Figure 9.7 "Trigger Transmit Request"

Adapt API Can_SetControllerMode() to new signature:
* Figure 9.11: Start CAN network
* Figure 9.13: BusOff recovery

Figure 9.13: Change typo "Cnange" to "Change"

~SWS_CANIF_00678: If all calls of Can_CheckWakeup() or CanTrcv_CheckWakeup() return E_NOT_OK to CanIf, then CanIf_CheckWakeup() shall return E_NOT_OK.

~SWS_CANIF_00720: If at least one function call of Can_CheckWakeup() or CanTrcv_CheckWakeup() returns E_OK to CanIf, then CanIf_CheckWakeup() shall return E_OK.

Rename CAN_OK to E_OK and CAN_NOT_OK to E_NOT_OK:
Note between SWS_CANIF_00162 and SWS_CANIF_00319
Table in chapter 9.7 Trigger Transmit Request
Table in chapter 9.11 Start CAN network

=== CanTrcv ===
Adapt API Can_SetControllerMode() to new signature:
* 9.3 De-Initialization (SPI Synchronous)
* 9.4 De-Initialization (SPI Asynchronous)

=== EcuSM ===
Adapt API Can_CheckWakeup() to new signature:
* Figure 42 CAN controller wake up by interrupt
* Figure 43 CAN controller or transceiver wake up by polling

=== TTCanIf ===
Adapt API Can_Write() to new signature:
* Figure 9.1: CAN Interface Time Triggered transmission with Job List

Correct API Can_TTReceive() which has return void instead of Can_ReturnType indeed:
* Figure 9.2: CAN Interface Time Triggered reception with Job List

Rename CAN_OK to E_OK and CAN_NOT_OK to E_NOT_OK:

~SWS_TtCanIf_00071

=== TTCanDrv ===
Rename CAN_OK to E_OK and CAN_NOT_OK to E_NOT_OK:
~SWS_TtCan_00014
~SWS_TtCan_00018
~SWS_TtCan_00022
~SWS_TtCan_00026
~SWS_TtCan_00059
~SWS_TtCan_00078
~SWS_TtCan_00112

=== XCP ===
Adapt API Can_Write() to new signature:
* Figure 5: Xcp on Can Transmit
–Last change on issue 77952 comment 22–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.16 Specification Item SWS_Can_00205

Trace References:

none

Content:

If default development error detection for the Can module is enabled: The function Can_DisableControllerInterrupts shall raise the error CAN_E_UNINIT if the driver not yet initialized.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #73570: No "default error" in AUTOSAR

  Problem description:

  The DET was renamed from development error tracer to default error tracer.

  This change was most of the time done automatically and unfortunately renamed "developement error" to "default error".

"default error" should always be followed by "tracer", otherwise, "development error" is probably the right term.

This could increase the impact (compared to my selection of impacted document, but formally, the configuration parameters *DevErrorDetect are not using the correct description:
"Switches the Default Error Tracer (Det) detection and notification..."

The parameter switches on/off the developement error detection. The DET does not need to be detected and can be present even when the parameter is set to false.

Agreed solution:

Rename "default error" to "development error" in all impacted documents, but not in an automated way (Do not change "default error tracer" to "developement error tracer"!)

Blueprint/Example:
- sub chapter is now called "7.x Default errors"
- "[SWS_xxx_yyyyy]
In case default error detection is enabled for the xxxx module: The xxxx module shall check API parameters for validity and report detected errors to the DET. ()"
- "[SWS_xxx_yyyyy]
If default error detection is enabled: the function shall check that the service xxx_Init was previously called. If the check fails, the function shall raise the default error XXX_E_NOT_INITIALIZED otherwise (if DET is disabled) return E_NOT_OK. ()"
- "In case default errors are enabled,..."
- "module raises the Default error XXX_E_TRANSITION"
- "The DET provides services to store default errors"
...

The correct text would be:
- sub chapter is called "7.x Development errors"
- "[SWS_xxx_yyyyy]
In case development error detection is enabled for the xxxx module: The xxxx module shall check API parameters for validity and report detected development errors to the DET. ()"
- "[SWS_xxx_yyyyy]
If development error detection is enabled: the function shall check that the service xxx_Init was previously called. If the check fails, the function shall raise the development error XXX_E_NOT_INITIALIZED otherwise (if DET is disabled) return E_NOT_OK. ()"

- "In case development errors are enabled,..."
- "module raises the development error XXX_E_TRANSITION"
- "The DET provides services to store development errors"

Solution for SWS_RTE:
– SWS_RTE —
- Change 4.8 Default errors to 4.8 Development errors
- Change "Errors which can occur at runtime in the RTE are classified as default errors" to "Errors which can occur at runtime in the RTE are classified as development errors"
- Remove [SWS_Rte_07676]
- Change [SWS_RTE_06611]"If a violation is detected the RTE shall report a default error to the DET." to "If a violation is detected the RTE shall report a development error to the DET."
- Change [SWS_Rte_06631]
[SWS_Rte_06631] d The RTE shall use the OS Application Identifier as the Instance Id to enable the developer to identify in which runtime section of the RTE the error occurs. This Instance ID is even unique across multi cores and so implicitly allows the development error to be traced to a specific core. c(SRS_BSW_00337)

SRS_Libraries:
- In chapter "3 Acronyms and abbreviations": Rename "Development Error Tracer" to "Default Error Tracer"

SRS_SPALGeneral:
- In chapter "6.1.1.3.1 [SRS_SPAL_00157] ...": Rename "Development Error Tracer" to "Default Error Tracer"
- In chapter "6.1.1.4.2 [SRS_SPAL_12448] ...": Rename "Development Error Tracer" to "Default Error Tracer"

SRS_FlashTest:
- In chapter "6.1 Functional Requirements": Rename "Development Error Tracer" to "Default Error Tracer"
- In chapter "7 References":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_MFXLibrary:

- In chapter "2 Acronyms and abbreviations": Rename "Development Error Tracer" to "Default Error Tracer"

SWS_MemoryAbstractionInterface:
- In chapter "3.1 Input documents":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_FlexRayNetworkManagement:
- In chapter "3.3 Related AUTOSAR documents":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_CANStateManager:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_PDURouter:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_EEPROMDriver:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"
–Last change on issue 73570 comment 47–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.17 Specification Item SWS_Can_00206

Trace References:

none

Content:

If default development error detection for the Can module is enabled: The function Can_DisableControllerInterrupts shall raise the error CAN_E_PARAM_CONTROLLER if the parameter Controller is out of range.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #73570: No "default error" in AUTOSAR

  Problem description:

  The DET was renamed from development error tracer to default error tracer.

  This change was most of the time done automatically and unfortunately renamed "developement error" to "default error".

  "default error" should always be followed by "tracer", otherwise, "development error" is probably the right term.

  This could increase the impact (compared to my selection of impacted document, but formally, the configuration parameters *DevErrorDetect are not using the correct description:
  "Switches the Default Error Tracer (Det) detection and notification..."

  The parameter switches on/off the developement error detection. The DET does not need to be detected and can be present even when the parameter is set to false.

  Agreed solution:

  Rename "default error" to "development error" in all impacted documents, but not in an automated way (Do not change "default error tracer" to "developement error tracer"!)

  Blueprint/Example:
  - sub chapter is now called "7.x Default errors"
  - "[SWS_xxx_yyyyy]
  In case default error detection is enabled for the xxxx module: The xxxx module shall check API parameters for validity and report detected errors to the DET. ()"
  - "[SWS_xxx_yyyyy]
  If default error detection is enabled: the function shall check that the service xxx_Init was previously called. If the check fails, the function shall raise the default error XXX_E_NOT_INITIALIZED otherwise (if DET is disabled) return E_NOT_OK. ()"
  - "In case default errors are enabled,..."

- "module raises the Default error XXX_E_TRANSITION"
- "The DET provides services to store default errors"
...

The correct text would be:
- sub chapter is called "7.x Development errors"
- "[SWS_xxx_yyyyy]
In case development error detection is enabled for the xxxx module: The xxxx module shall check API parameters for validity and report detected development errors to the DET. ()"
- "[SWS_xxx_yyyyy]
If development error detection is enabled: the function shall check that the service xxx_Init was previously called. If the check fails, the function shall raise the development error XXX_E_NOT_INITIALIZED otherwise (if DET is disabled) return E_NOT_OK. ()"
- "In case development errors are enabled,..."
- "module raises the development error XXX_E_TRANSITION"
- "The DET provides services to store development errors"

Solution for SWS_RTE:
– SWS_RTE —
- Change 4.8 Default errors to 4.8 Development errors
- Change "Errors which can occur at runtime in the RTE are classified as default errors" to "Errors which can occur at runtime in the RTE are classified as development errors"
- Remove [SWS_Rte_07676]
- Change [SWS_RTE_06611]"If a violation is detected the RTE shall report a default error to the DET." to "If a violation is detected the RTE shall report a development error to the DET."
- Change [SWS_Rte_06631]
[SWS_Rte_06631] d The RTE shall use the OS Application Identifier as the Instance Id to enable the developer to identify in which runtime section of the RTE the error occurs. This Instance ID is even unique across multi cores and so implicitly allows the development error to be traced to a specific core. c(SRS_BSW_00337)

SRS_Libraries:
- In chapter "3 Acronyms and abbreviations": Rename "Development Error Tracer" to "Default Error Tracer"

SRS_SPALGeneral:

- In chapter "6.1.1.3.1 [SRS_SPAL_00157] ...": Rename "Development Error Tracer" to "Default Error Tracer"
- In chapter "6.1.1.4.2 [SRS_SPAL_12448] ...": Rename "Development Error Tracer" to "Default Error Tracer"

SRS_FlashTest:
- In chapter "6.1 Functional Requirements": Rename "Development Error Tracer" to "Default Error Tracer"
- In chapter "7 References":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_MFXLibrary:
- In chapter "2 Acronyms and abbreviations": Rename "Development Error Tracer" to "Default Error Tracer"

SWS_MemoryAbstractionInterface:
- In chapter "3.1 Input documents":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_FlexRayNetworkManagement:
- In chapter "3.3 Related AUTOSAR documents":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_CANStateManager:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_PDURouter:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_EEPROMDriver:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"
–Last change on issue 73570 comment 47–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.18 Specification Item SWS_Can_00209

Trace References:

none

Content:

If default development error detection for the Can module is enabled: The function Can_EnableControllerInterrupts shall raise the error CAN_E_UNINIT if the driver not yet initialized.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #73570: No "default error" in AUTOSAR

  Problem description:

  The DET was renamed from development error tracer to default error tracer.

  This change was most of the time done automatically and unfortunately renamed "developement error" to "default error".

  "default error" should always be followed by "tracer", otherwise, "development error" is probably the right term.

  This could increase the impact (compared to my selection of impacted document, but formally, the configuration parameters *DevErrorDetect are not using the correct description:
  "Switches the Default Error Tracer (Det) detection and notification..."

  The parameter switches on/off the developement error detection. The DET does not need to be detected and can be present even when the parameter is set to false.

Agreed solution:

Rename "default error" to "development error" in all impacted documents, but not in an automated way (Do not change "default error tracer" to "developement error tracer"!)

Blueprint/Example:
- sub chapter is now called "7.x Default errors"
- "[SWS_xxx_yyyyy]
In case default error detection is enabled for the xxxx module: The xxxx module shall check API parameters for validity and report detected errors to the DET. ()"
- "[SWS_xxx_yyyyy]
If default error detection is enabled: the function shall check that the service xxx_Init was previously called. If the check fails, the function shall raise the default error XXX_E_NOT_INITIALIZED otherwise (if DET is disabled) return E_NOT_OK. ()"
- "In case default errors are enabled,..."
- "module raises the Default error XXX_E_TRANSITION"
- "The DET provides services to store default errors"
...

The correct text would be:
- sub chapter is called "7.x Development errors"
- "[SWS_xxx_yyyyy]
In case development error detection is enabled for the xxxx module: The xxxx module shall check API parameters for validity and report detected development errors to the DET. ()"
- "[SWS_xxx_yyyyy]
If development error detection is enabled: the function shall check that the service xxx_Init was previously called. If the check fails, the function shall raise the development error XXX_E_NOT_INITIALIZED otherwise (if DET is disabled) return E_NOT_OK. ()"
- "In case development errors are enabled,..."
- "module raises the development error XXX_E_TRANSITION"
- "The DET provides services to store development errors"

Solution for SWS_RTE:
– SWS_RTE —
- Change 4.8 Default errors to 4.8 Development errors
- Change "Errors which can occur at runtime in the RTE are classified as default errors" to "Errors which can occur at runtime in the RTE are classified as development errors"
- Remove [SWS_Rte_07676]

- Change [SWS_RTE_06611]"If a violation is detected the RTE shall report a default error to the DET." to "If a violation is detected the RTE shall report a development error to the DET."
- Change [SWS_Rte_06631]
[SWS_Rte_06631] d The RTE shall use the OS Application Identifier as the Instance Id to enable the developer to identify in which runtime section of the RTE the error occurs. This Instance ID is even unique across multi cores and so implicitly allows the development error to be traced to a specific core. c(SRS_BSW_00337)

SRS_Libraries:
- In chapter "3 Acronyms and abbreviations": Rename "Development Error Tracer" to "Default Error Tracer"

SRS_SPALGeneral:
- In chapter "6.1.1.3.1 [SRS_SPAL_00157] ...": Rename "Development Error Tracer" to "Default Error Tracer"
- In chapter "6.1.1.4.2 [SRS_SPAL_12448] ...": Rename "Development Error Tracer" to "Default Error Tracer"

SRS_FlashTest:
- In chapter "6.1 Functional Requirements": Rename "Development Error Tracer" to "Default Error Tracer"
- In chapter "7 References":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_MFXLibrary:
- In chapter "2 Acronyms and abbreviations": Rename "Development Error Tracer" to "Default Error Tracer"

SWS_MemoryAbstractionInterface:
- In chapter "3.1 Input documents":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_FlexRayNetworkManagement:
- In chapter "3.3 Related AUTOSAR documents":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_CANStateManager:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_PDURouter:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_EEPROMDriver:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"
–Last change on issue 73570 comment 47–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.19 Specification Item SWS_Can_00210

Trace References:

none

Content:

If default development error detection for the Can module is enabled: The function Can_EnableControllerInterrupts shall raise the error CAN_E_PARAM_CONTROLLER if the parameter Controller is out of range.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #73570: No "default error" in AUTOSAR

  Problem description:

The DET was renamed from development error tracer to default error tracer.

This change was most of the time done automatically and unfortunately renamed "developement error" to "default error".

"default error" should always be followed by "tracer", otherwise, "development error" is probably the right term.

This could increase the impact (compared to my selection of impacted document, but formally, the configuration parameters *DevErrorDetect are not using the correct description:
"Switches the Default Error Tracer (Det) detection and notification..."

The parameter switches on/off the developement error detection. The DET does not need to be detected and can be present even when the parameter is set to false.

Agreed solution:

Rename "default error" to "development error" in all impacted documents, but not in an automated way (Do not change "default error tracer" to "developement error tracer"!)

Blueprint/Example:
- sub chapter is now called "7.x Default errors"
- "[SWS_xxx_yyyyy]
In case default error detection is enabled for the xxxx module: The xxxx module shall check API parameters for validity and report detected errors to the DET. ()"
- "[SWS_xxx_yyyyy]
If default error detection is enabled: the function shall check that the service xxx_Init was previously called. If the check fails, the function shall raise the default error XXX_E_NOT_INITIALIZED otherwise (if DET is disabled) return E_NOT_OK. ()"
- "In case default errors are enabled,..."
- "module raises the Default error XXX_E_TRANSITION"
- "The DET provides services to store default errors"
...

The correct text would be:
- sub chapter is called "7.x Development errors"
- "[SWS_xxx_yyyyy]
In case development error detection is enabled for the xxxx module: The xxxx module shall check API parameters for validity and report detected development errors to the DET. ()"

- "[SWS_xxx_yyyyy]
If development error detection is enabled: the function shall check that the service xxx_Init was previously called. If the check fails, the function shall raise the development error XXX_E_NOT_INITIALIZED otherwise (if DET is disabled) return E_NOT_OK. ()"
- "In case development errors are enabled,..."
- "module raises the development error XXX_E_TRANSITION"
- "The DET provides services to store development errors"

Solution for SWS_RTE:
– SWS_RTE —
- Change 4.8 Default errors to 4.8 Development errors
- Change "Errors which can occur at runtime in the RTE are classified as default errors" to "Errors which can occur at runtime in the RTE are classified as development errors"
- Remove [SWS_Rte_07676]
- Change [SWS_RTE_06611]"If a violation is detected the RTE shall report a default error to the DET." to "If a violation is detected the RTE shall report a development error to the DET."
- Change [SWS_Rte_06631]
[SWS_Rte_06631] d The RTE shall use the OS Application Identifier as the Instance Id to enable the developer to identify in which runtime section of the RTE the error occurs. This Instance ID is even unique across multi cores and so implicitly allows the development error to be traced to a specific core. c(SRS_BSW_00337)

SRS_Libraries:
- In chapter "3 Acronyms and abbreviations": Rename "Development Error Tracer" to "Default Error Tracer"

SRS_SPALGeneral:
- In chapter "6.1.1.3.1 [SRS_SPAL_00157] ...": Rename "Development Error Tracer" to "Default Error Tracer"
- In chapter "6.1.1.4.2 [SRS_SPAL_12448] ...": Rename "Development Error Tracer" to "Default Error Tracer"

SRS_FlashTest:
- In chapter "6.1 Functional Requirements": Rename "Development Error Tracer" to "Default Error Tracer"
- In chapter "7 References":
Rename "Development Error Tracer" to "Default Error Tracer"

Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_MFXLibrary:
- In chapter "2 Acronyms and abbreviations": Rename "Development Error Tracer" to "Default Error Tracer"

SWS_MemoryAbstractionInterface:
- In chapter "3.1 Input documents":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_FlexRayNetworkManagement:
- In chapter "3.3 Related AUTOSAR documents":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_CANStateManager:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_PDURouter:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_EEPROMDriver:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"
–Last change on issue 73570 comment 47–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.20 Specification Item SWS_Can_00212

Trace References:

SRS_Can_01049

Content:

The function Can_Write shall perform following actions if the hardware transmit object is free:

- The mutex for that HTH is set to 'signaled'
- The ID, Data Length and SDU are put in a format appropriate for the hardware (if necessary) and copied in the appropriate hardware registers/buffers.
- All necessary control operations to initiate the transmit are done
- The mutex for that HTH is released
- The function returns with CANE_OK

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77952: [Can][CanIf] Incompatible return types of Can and CanTrcv

  Problem description:

  While CanTrcv only uses Std_ReturnType, Can uses Can_ReturnType in many places, even when only CAN_OK and CAN_NOT_OK are available.

  This leads to complicated code in CanIf, because it needs to implement separate checks for return values from CanTrcv and Can and cannot just combine the results.

  Agreed solution:

  === CanDrv ===

  Change of SWS_Can_00039 Can_ReturnType:
  * type change from enumeration to extra_litaral
  * Remove range element CAN_OK
  * Remove range element CAN_NOT_OK
  * Assign value "0x02" to range element "CAN_BUSY"
  * Description: Overlayed return value of Std_ReturnType for CAN driver API Can_Write().

  ~SWS_Can_00230 Can_SetControllerMode

Syntax: Std_ReturnType Can_SetControllerMode(uint8 Controller, Can_StateTransitionType Transition)
Return value:
Std_ReturnType
E_OK: request accepted
E_NOT_OK: request not accepted, a development error occurred

~SWS_Can_00360 Can_CheckWakeup
Syntax: Std_ReturnType Can_CheckWakeup(uint8 Controller)
Return value:
Std_ReturnType
E_OK: API call has been accepted
E_NOT_OK: API call has not been accepted

~SWS_Can_00233 Can_Write
Syntax: Std_ReturnType Can_Write(Can_HwHandleType Hth, const Can_PduType* PduInfo)
Return value:
Std_ReturnType
E_OK: Write command has been accepted
E_NOT_OK: development error occurred
CAN_BUSY: No TX hardware buffer available or pre-emptive call of Can_Write that can't be implemented re-entrant (see Can_ReturnType)

Rename CAN_OK to E_OK and CAN_NOT_OK to E_NOT_OK:
~SWS_Can_00048
~SWS_Can_00089
7.11.5 Return Values
~SWS_Can_00198
~SWS_Can_00199
~SWS_Can_00200
~SWS_Can_00216
~SWS_Can_00217
~SWS_Can_00218
~SWS_CAN_00219
~SWS_CAN_00505
~SWS_CAN_00506
~SWS_Can_00212

=== CanIf ===
Adapt API Can_Write() to new signature:
* Figure 7.10 "Transmission request with multiple CAN Drivers - simplified"
* Figure 9.1 "Transmission request with a single CAN Driver"

* Figure 9.2 "Transmission request with multiple CAN Drivers"
* Figure 9.5 "Transmit confirmation with buffering"
* Figure 9.6 "Transmit Cancelation"
* Figure 9.7 "Trigger Transmit Request"

Adapt API Can_SetControllerMode() to new signature:
* Figure 9.11: Start CAN network
* Figure 9.13: BusOff recovery

Figure 9.13: Change typo "Cnange" to "Change"

~SWS_CANIF_00678: If all calls of Can_CheckWakeup() or CanTrcv_CheckWakeup() return E_NOT_OK to CanIf, then CanIf_CheckWakeup() shall return E_NOT_OK.

~SWS_CANIF_00720: If at least one function call of Can_CheckWakeup() or CanTrcv_CheckWakeup() returns E_OK to CanIf, then CanIf_CheckWakeup() shall return E_OK.

Rename CAN_OK to E_OK and CAN_NOT_OK to E_NOT_OK:
Note between SWS_CANIF_00162 and SWS_CANIF_00319
Table in chapter 9.7 Trigger Transmit Request
Table in chapter 9.11 Start CAN network

=== CanTrcv ===
Adapt API Can_SetControllerMode() to new signature:
* 9.3 De-Initialization (SPI Synchronous)
* 9.4 De-Initialization (SPI Asynchronous)

=== EcuSM ===
Adapt API Can_CheckWakeup() to new signature:
* Figure 42 CAN controller wake up by interrupt
* Figure 43 CAN controller or transceiver wake up by polling

=== TTCanIf ===
Adapt API Can_Write() to new signature:
* Figure 9.1: CAN Interface Time Triggered transmission with Job List

Correct API Can_TTReceive() which has return void instead of Can_ReturnType indeed:
* Figure 9.2: CAN Interface Time Triggered reception with Job List

Rename CAN_OK to E_OK and CAN_NOT_OK to E_NOT_OK:

~SWS_TtCanIf_00071

=== TTCanDrv ===
Rename CAN_OK to E_OK and CAN_NOT_OK to E_NOT_OK:
~SWS_TtCan_00014
~SWS_TtCan_00018
~SWS_TtCan_00022
~SWS_TtCan_00026
~SWS_TtCan_00059
~SWS_TtCan_00078
~SWS_TtCan_00112

=== XCP ===
Adapt API Can_Write() to new signature:
* Figure 5: Xcp on Can Transmit
–Last change on issue 77952 comment 22–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.21 Specification Item SWS_Can_00216

Trace References:

none

Content:

If default development error detection for the Can module is enabled: The function Can_Write shall raise the error CAN_E_UNINIT and shall return CANE_NOT_OK if the driver is not yet initialized.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #73570: No "default error" in AUTOSAR

  Problem description:

  The DET was renamed from development error tracer to default error tracer.

  This change was most of the time done automatically and unfortunately renamed "developement error" to "default error".

"default error" should always be followed by "tracer", otherwise, "development error" is probably the right term.

This could increase the impact (compared to my selection of impacted document, but formally, the configuration parameters *DevErrorDetect are not using the correct description:
"Switches the Default Error Tracer (Det) detection and notification..."

The parameter switches on/off the developement error detection. The DET does not need to be detected and can be present even when the parameter is set to false.

Agreed solution:

Rename "default error" to "development error" in all impacted documents, but not in an automated way (Do not change "default error tracer" to "developement error tracer"!)

Blueprint/Example:
- sub chapter is now called "7.x Default errors"
- "[SWS_xxx_yyyyy]
In case default error detection is enabled for the xxxx module: The xxxx module shall check API parameters for validity and report detected errors to the DET. ()"
- "[SWS_xxx_yyyyy]
If default error detection is enabled: the function shall check that the service xxx_Init was previously called. If the check fails, the function shall raise the default error XXX_E_NOT_INITIALIZED otherwise (if DET is disabled) return E_NOT_OK. ()"
- "In case default errors are enabled,..."
- "module raises the Default error XXX_E_TRANSITION"
- "The DET provides services to store default errors"
...

The correct text would be:
- sub chapter is called "7.x Development errors"
- "[SWS_xxx_yyyyy]
In case development error detection is enabled for the xxxx module: The xxxx module shall check API parameters for validity and report detected development errors to the DET. ()"
- "[SWS_xxx_yyyyy]
If development error detection is enabled: the function shall check that the service xxx_Init was previously called. If the check fails, the function shall raise the development error XXX_E_NOT_INITIALIZED otherwise (if DET is disabled) return E_NOT_OK. ()"

- "In case development errors are enabled,..."
- "module raises the development error XXX_E_TRANSITION"
- "The DET provides services to store development errors"

Solution for SWS_RTE:
– SWS_RTE —
- Change 4.8 Default errors to 4.8 Development errors
- Change "Errors which can occur at runtime in the RTE are classified as default errors" to "Errors which can occur at runtime in the RTE are classified as development errors"
- Remove [SWS_Rte_07676]
- Change [SWS_RTE_06611]"If a violation is detected the RTE shall report a default error to the DET." to "If a violation is detected the RTE shall report a development error to the DET."
- Change [SWS_Rte_06631]
[SWS_Rte_06631] d The RTE shall use the OS Application Identifier as the Instance Id to enable the developer to identify in which runtime section of the RTE the error occurs. This Instance ID is even unique across multi cores and so implicitly allows the development error to be traced to a specific core. c(SRS_BSW_00337)

SRS_Libraries:
- In chapter "3 Acronyms and abbreviations": Rename "Development Error Tracer" to "Default Error Tracer"

SRS_SPALGeneral:
- In chapter "6.1.1.3.1 [SRS_SPAL_00157] ...": Rename "Development Error Tracer" to "Default Error Tracer"
- In chapter "6.1.1.4.2 [SRS_SPAL_12448] ...": Rename "Development Error Tracer" to "Default Error Tracer"

SRS_FlashTest:
- In chapter "6.1 Functional Requirements": Rename "Development Error Tracer" to "Default Error Tracer"
- In chapter "7 References":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_MFXLibrary:

- In chapter "2 Acronyms and abbreviations": Rename "Development Error Tracer" to "Default Error Tracer"

SWS_MemoryAbstractionInterface:
- In chapter "3.1 Input documents":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_FlexRayNetworkManagement:
- In chapter "3.3 Related AUTOSAR documents":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_CANStateManager:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_PDURouter:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_EEPROMDriver:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"
–Last change on issue 73570 comment 47–

BW-C-Level:

Application	Specification	Bus
1	1	1

- RfC #77952: [Can][CanIf] Incompatible return types of Can and CanTrcv

  Problem description:

  While CanTrcv only uses Std_ReturnType, Can uses Can_ReturnType in many places, even when only CAN_OK and CAN_NOT_OK are available.

This leads to complicated code in CanIf, because it needs to implement separate checks for return values from CanTrcv and Can and cannot just combine the results.

Agreed solution:

=== CanDrv ===

Change of SWS_Can_00039 Can_ReturnType:
* type change from enumeration to extra_litaral
* Remove range element CAN_OK
* Remove range element CAN_NOT_OK
* Assign value "0x02" to range element "CAN_BUSY"
* Description: Overlayed return value of Std_ReturnType for CAN driver API Can_Write().

~SWS_Can_00230 Can_SetControllerMode
Syntax: Std_ReturnType Can_SetControllerMode(uint8 Controller, Can_StateTransitionType Transition)
Return value:
Std_ReturnType
E_OK: request accepted
E_NOT_OK: request not accepted, a development error occurred

~SWS_Can_00360 Can_CheckWakeup
Syntax: Std_ReturnType Can_CheckWakeup(uint8 Controller)
Return value:
Std_ReturnType
E_OK: API call has been accepted
E_NOT_OK: API call has not been accepted

~SWS_Can_00233 Can_Write
Syntax: Std_ReturnType Can_Write(Can_HwHandleType Hth, const Can_PduType* PduInfo)
Return value:
Std_ReturnType
E_OK: Write command has been accepted
E_NOT_OK: development error occurred
CAN_BUSY: No TX hardware buffer available or pre-emptive call of Can_Write that can't be implemented re-entrant (see Can_ReturnType)

Rename CAN_OK to E_OK and CAN_NOT_OK to E_NOT_OK:
~SWS_Can_00048
~SWS_Can_00089

7.11.5 Return Values
~SWS_Can_00198
~SWS_Can_00199
~SWS_Can_00200
~SWS_Can_00216
~SWS_Can_00217
~SWS_Can_00218
~SWS_CAN_00219
~SWS_CAN_00505
~SWS_CAN_00506
~SWS_Can_00212

=== CanIf ===
Adapt API Can_Write() to new signature:
* Figure 7.10 "Transmission request with multiple CAN Drivers - simplified"
* Figure 9.1 "Transmission request with a single CAN Driver"
* Figure 9.2 "Transmission request with multiple CAN Drivers"
* Figure 9.5 "Transmit confirmation with buffering"
* Figure 9.6 "Transmit Cancelation"
* Figure 9.7 "Trigger Transmit Request"

Adapt API Can_SetControllerMode() to new signature:
* Figure 9.11: Start CAN network
* Figure 9.13: BusOff recovery

Figure 9.13: Change typo "Cnange" to "Change"

~SWS_CANIF_00678: If all calls of Can_CheckWakeup() or CanTrcv_CheckWakeup() return E_NOT_OK to CanIf, then CanIf_CheckWakeup() shall return E_NOT_OK.

~SWS_CANIF_00720: If at least one function call of Can_CheckWakeup() or CanTrcv_CheckWakeup() returns E_OK to CanIf, then CanIf_CheckWakeup() shall return E_OK.

Rename CAN_OK to E_OK and CAN_NOT_OK to E_NOT_OK:
Note between SWS_CANIF_00162 and SWS_CANIF_00319
Table in chapter 9.7 Trigger Transmit Request
Table in chapter 9.11 Start CAN network

=== CanTrcv ===
Adapt API Can_SetControllerMode() to new signature:
* 9.3 De-Initialization (SPI Synchronous)

* 9.4 De-Initialization (SPI Asynchronous)

=== EcuSM ===
Adapt API Can_CheckWakeup() to new signature:
* Figure 42 CAN controller wake up by interrupt
* Figure 43 CAN controller or transceiver wake up by polling

=== TTCanIf ===
Adapt API Can_Write() to new signature:
* Figure 9.1: CAN Interface Time Triggered transmission with Job List

Correct API Can_TTReceive() which has return void instead of Can_ReturnType indeed:
* Figure 9.2: CAN Interface Time Triggered reception with Job List

Rename CAN_OK to E_OK and CAN_NOT_OK to E_NOT_OK:
~SWS_TtCanIf_00071

=== TTCanDrv ===
Rename CAN_OK to E_OK and CAN_NOT_OK to E_NOT_OK:
~SWS_TtCan_00014
~SWS_TtCan_00018
~SWS_TtCan_00022
~SWS_TtCan_00026
~SWS_TtCan_00059
~SWS_TtCan_00078
~SWS_TtCan_00112

=== XCP ===
Adapt API Can_Write() to new signature:
* Figure 5: Xcp on Can Transmit
–Last change on issue 77952 comment 22–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.22 Specification Item SWS_Can_00217

Trace References:

none

Content:

If default development error detection for the Can module is enabled: The function Can_Write shall raise the error CAN_E_PARAM_HANDLE and shall return CANE_NOT_OK if the parameter Hth is not a configured Hardware Transmit Handle.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #73570: No "default error" in AUTOSAR

  Problem description:

  The DET was renamed from development error tracer to default error tracer.

  This change was most of the time done automatically and unfortunately renamed "developement error" to "default error".

  "default error" should always be followed by "tracer", otherwise, "development error" is probably the right term.

  This could increase the impact (compared to my selection of impacted document, but formally, the configuration parameters *DevErrorDetect are not using the correct description:
  "Switches the Default Error Tracer (Det) detection and notification..."

  The parameter switches on/off the developement error detection. The DET does not need to be detected and can be present even when the parameter is set to false.

  Agreed solution:

  Rename "default error" to "development error" in all impacted documents, but not in an automated way (Do not change "default error tracer" to "developement error tracer"!)

  Blueprint/Example:
  - sub chapter is now called "7.x Default errors"
  - "[SWS_xxx_yyyyy]
  In case default error detection is enabled for the xxxx module: The xxxx module shall check API parameters for validity and report detected errors to the DET. ()"
  - "[SWS_xxx_yyyyy]
  If default error detection is enabled: the function shall check that the service xxx_Init was previously called. If the check fails, the function shall raise the default error XXX_E_NOT_INITIALIZED otherwise (if DET is disabled) return E_NOT_OK. ()"
  - "In case default errors are enabled,..."

- "module raises the Default error XXX_E_TRANSITION"
- "The DET provides services to store default errors"
...

The correct text would be:
- sub chapter is called "7.x Development errors"
- "[SWS_xxx_yyyyy]
In case development error detection is enabled for the xxxx module: The xxxx module shall check API parameters for validity and report detected development errors to the DET. ()"
- "[SWS_xxx_yyyyy]
If development error detection is enabled: the function shall check that the service xxx_Init was previously called. If the check fails, the function shall raise the development error XXX_E_NOT_INITIALIZED otherwise (if DET is disabled) return E_NOT_OK. ()"
- "In case development errors are enabled,..."
- "module raises the development error XXX_E_TRANSITION"
- "The DET provides services to store development errors"

Solution for SWS_RTE:
– SWS_RTE —
- Change 4.8 Default errors to 4.8 Development errors
- Change "Errors which can occur at runtime in the RTE are classified as default errors" to "Errors which can occur at runtime in the RTE are classified as development errors"
- Remove [SWS_Rte_07676]
- Change [SWS_RTE_06611]"If a violation is detected the RTE shall report a default error to the DET." to "If a violation is detected the RTE shall report a development error to the DET."
- Change [SWS_Rte_06631]
[SWS_Rte_06631] d The RTE shall use the OS Application Identifier as the Instance Id to enable the developer to identify in which runtime section of the RTE the error occurs. This Instance ID is even unique across multi cores and so implicitly allows the development error to be traced to a specific core. c(SRS_BSW_00337)

SRS_Libraries:
- In chapter "3 Acronyms and abbreviations": Rename "Development Error Tracer" to "Default Error Tracer"

SRS_SPALGeneral:

- In chapter "6.1.1.3.1 [SRS_SPAL_00157] ...": Rename "Development Error Tracer" to "Default Error Tracer"
- In chapter "6.1.1.4.2 [SRS_SPAL_12448] ...": Rename "Development Error Tracer" to "Default Error Tracer"

SRS_FlashTest:
- In chapter "6.1 Functional Requirements": Rename "Development Error Tracer" to "Default Error Tracer"
- In chapter "7 References":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_MFXLibrary:
- In chapter "2 Acronyms and abbreviations": Rename "Development Error Tracer" to "Default Error Tracer"

SWS_MemoryAbstractionInterface:
- In chapter "3.1 Input documents":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_FlexRayNetworkManagement:
- In chapter "3.3 Related AUTOSAR documents":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_CANStateManager:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_PDURouter:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_EEPROMDriver:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"
–Last change on issue 73570 comment 47–

BW-C-Level:

Application	Specification	Bus
1	1	1

- RfC #77952: [Can][CanIf] Incompatible return types of Can and CanTrcv

  Problem description:

  While CanTrcv only uses Std_ReturnType, Can uses Can_ReturnType in many places, even when only CAN_OK and CAN_NOT_OK are available.

  This leads to complicated code in CanIf, because it needs to implement separate checks for return values from CanTrcv and Can and cannot just combine the results.

  Agreed solution:

  === CanDrv ===

  Change of SWS_Can_00039 Can_ReturnType:
  * type change from enumeration to extra_litaral
  * Remove range element CAN_OK
  * Remove range element CAN_NOT_OK
  * Assign value "0x02" to range element "CAN_BUSY"
  * Description: Overlayed return value of Std_ReturnType for CAN driver API Can_Write().

  ~SWS_Can_00230 Can_SetControllerMode
  Syntax: Std_ReturnType Can_SetControllerMode(uint8 Controller, Can_StateTransitionType Transition)
  Return value:
  Std_ReturnType
  E_OK: request accepted
  E_NOT_OK: request not accepted, a development error occurred

  ~SWS_Can_00360 Can_CheckWakeup
  Syntax: Std_ReturnType Can_CheckWakeup(uint8 Controller)
  Return value:
  Std_ReturnType

E_OK: API call has been accepted
E_NOT_OK: API call has not been accepted

~SWS_Can_00233 Can_Write
Syntax: Std_ReturnType Can_Write(Can_HwHandleType Hth, const Can_PduType* PduInfo)
Return value:
Std_ReturnType
E_OK: Write command has been accepted
E_NOT_OK: development error occurred
CAN_BUSY: No TX hardware buffer available or pre-emptive call of Can_Write that can't be implemented re-entrant (see Can_ReturnType)

Rename CAN_OK to E_OK and CAN_NOT_OK to E_NOT_OK:
~SWS_Can_00048
~SWS_Can_00089
7.11.5 Return Values
~SWS_Can_00198
~SWS_Can_00199
~SWS_Can_00200
~SWS_Can_00216
~SWS_Can_00217
~SWS_Can_00218
~SWS_CAN_00219
~SWS_CAN_00505
~SWS_CAN_00506
~SWS_Can_00212

=== CanIf ===
Adapt API Can_Write() to new signature:
* Figure 7.10 "Transmission request with multiple CAN Drivers - simplified"
* Figure 9.1 "Transmission request with a single CAN Driver"
* Figure 9.2 "Transmission request with multiple CAN Drivers"
* Figure 9.5 "Transmit confirmation with buffering"
* Figure 9.6 "Transmit Cancelation"
* Figure 9.7 "Trigger Transmit Request"

Adapt API Can_SetControllerMode() to new signature:
* Figure 9.11: Start CAN network
* Figure 9.13: BusOff recovery

Figure 9.13: Change typo "Cnange" to "Change"

~SWS_CANIF_00678: If all calls of Can_CheckWakeup() or CanTrcv_CheckWakeup() return E_NOT_OK to CanIf, then CanIf_CheckWakeup() shall return E_NOT_OK.

~SWS_CANIF_00720: If at least one function call of Can_CheckWakeup() or CanTrcv_CheckWakeup() returns E_OK to CanIf, then CanIf_CheckWakeup() shall return E_OK.

Rename CAN_OK to E_OK and CAN_NOT_OK to E_NOT_OK:
Note between SWS_CANIF_00162 and SWS_CANIF_00319
Table in chapter 9.7 Trigger Transmit Request
Table in chapter 9.11 Start CAN network

=== CanTrcv ===
Adapt API Can_SetControllerMode() to new signature:
* 9.3 De-Initialization (SPI Synchronous)
* 9.4 De-Initialization (SPI Asynchronous)

=== EcuSM ===
Adapt API Can_CheckWakeup() to new signature:
* Figure 42 CAN controller wake up by interrupt
* Figure 43 CAN controller or transceiver wake up by polling

=== TTCanIf ===
Adapt API Can_Write() to new signature:
* Figure 9.1: CAN Interface Time Triggered transmission with Job List

Correct API Can_TTReceive() which has return void instead of Can_ReturnType indeed:
* Figure 9.2: CAN Interface Time Triggered reception with Job List

Rename CAN_OK to E_OK and CAN_NOT_OK to E_NOT_OK:
~SWS_TtCanIf_00071

=== TTCanDrv ===
Rename CAN_OK to E_OK and CAN_NOT_OK to E_NOT_OK:
~SWS_TtCan_00014
~SWS_TtCan_00018
~SWS_TtCan_00022
~SWS_TtCan_00026
~SWS_TtCan_00059
~SWS_TtCan_00078
~SWS_TtCan_00112

=== XCP ===
Adapt API Can_Write() to new signature:
* Figure 5: Xcp on Can Transmit
–Last change on issue 77952 comment 22–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.23 Specification Item SWS_Can_00218

Trace References:

SRS_Can_01005

Content:

The function Can_Write shall return CANE_NOT_OK and if default development error detection for the CAN module is enabled shall raise the error CAN_E_PARAM_DATA_LENGTH:

- If the length is more than 64 byte.
- If the length is more than 8 byte and the CAN controller is not in CAN FD mode (no CanControllerFdBaudrateConfig).
- If the length is more than 8 byte and the CAN controller is in CAN FD mode (valid CanControllerFdBaudrateConfig), but the CAN FD flag in Can_PduType->id is not set (refer Can_IdType).

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #73570: No "default error" in AUTOSAR

  Problem description:

  The DET was renamed from development error tracer to default error tracer.

  This change was most of the time done automatically and unfortunately renamed "developement error" to "default error".

  "default error" should always be followed by "tracer", otherwise, "development error" is probably the right term.

This could increase the impact (compared to my selection of impacted document, but formally, the configuration parameters *DevErrorDetect are not using the correct description:
"Switches the Default Error Tracer (Det) detection and notification..."

The parameter switches on/off the developement error detection. The DET does not need to be detected and can be present even when the parameter is set to false.

Agreed solution:

Rename "default error" to "development error" in all impacted documents, but not in an automated way (Do not change "default error tracer" to "developement error tracer"!)

Blueprint/Example:
- sub chapter is now called "7.x Default errors"
- "[SWS_xxx_yyyyy]
In case default error detection is enabled for the xxxx module: The xxxx module shall check API parameters for validity and report detected errors to the DET. ()"
- "[SWS_xxx_yyyyy]
If default error detection is enabled: the function shall check that the service xxx_Init was previously called. If the check fails, the function shall raise the default error XXX_E_NOT_INITIALIZED otherwise (if DET is disabled) return E_NOT_OK. ()"
- "In case default errors are enabled,..."
- "module raises the Default error XXX_E_TRANSITION"
- "The DET provides services to store default errors"
...

The correct text would be:
- sub chapter is called "7.x Development errors"
- "[SWS_xxx_yyyyy]
In case development error detection is enabled for the xxxx module: The xxxx module shall check API parameters for validity and report detected development errors to the DET. ()"
- "[SWS_xxx_yyyyy]
If development error detection is enabled: the function shall check that the service xxx_Init was previously called. If the check fails, the function shall raise the development error XXX_E_NOT_INITIALIZED otherwise (if DET is disabled) return E_NOT_OK. ()"
- "In case development errors are enabled,..."
- "module raises the development error XXX_E_TRANSITION"
- "The DET provides services to store development errors"

Solution for SWS_RTE:
– SWS_RTE —
- Change 4.8 Default errors to 4.8 Development errors
- Change "Errors which can occur at runtime in the RTE are classified as default errors" to "Errors which can occur at runtime in the RTE are classified as development errors"
- Remove [SWS_Rte_07676]
- Change [SWS_RTE_06611]"If a violation is detected the RTE shall report a default error to the DET." to "If a violation is detected the RTE shall report a development error to the DET."
- Change [SWS_Rte_06631]
[SWS_Rte_06631] d The RTE shall use the OS Application Identifier as the Instance Id to enable the developer to identify in which runtime section of the RTE the error occurs. This Instance ID is even unique across multi cores and so implicitly allows the development error to be traced to a specific core. c(SRS_BSW_00337)

SRS_Libraries:
- In chapter "3 Acronyms and abbreviations": Rename "Development Error Tracer" to "Default Error Tracer"

SRS_SPALGeneral:
- In chapter "6.1.1.3.1 [SRS_SPAL_00157] ...": Rename "Development Error Tracer" to "Default Error Tracer"
- In chapter "6.1.1.4.2 [SRS_SPAL_12448] ...": Rename "Development Error Tracer" to "Default Error Tracer"

SRS_FlashTest:
- In chapter "6.1 Functional Requirements": Rename "Development Error Tracer" to "Default Error Tracer"
- In chapter "7 References":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_MFXLibrary:
- In chapter "2 Acronyms and abbreviations": Rename "Development Error Tracer" to "Default Error Tracer"

SWS_MemoryAbstractionInterface:
- In chapter "3.1 Input documents":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_FlexRayNetworkManagement:
- In chapter "3.3 Related AUTOSAR documents":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_CANStateManager:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_PDURouter:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_EEPROMDriver:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"
–Last change on issue 73570 comment 47–

BW-C-Level:

Application	Specification	Bus
1	1	1

- RfC #77952: [Can][CanIf] Incompatible return types of Can and CanTrcv

  Problem description:

  While CanTrcv only uses Std_ReturnType, Can uses Can_ReturnType in many places, even when only CAN_OK and CAN_NOT_OK are available.

  This leads to complicated code in CanIf, because it needs to implement separate checks for return values from CanTrcv and Can and cannot just combine the results.

Agreed solution:

=== CanDrv ===

Change of SWS_Can_00039 Can_ReturnType:
* type change from enumeration to extra_litaral
* Remove range element CAN_OK
* Remove range element CAN_NOT_OK
* Assign value "0x02" to range element "CAN_BUSY"
* Description: Overlayed return value of Std_ReturnType for CAN driver API Can_Write().

~SWS_Can_00230 Can_SetControllerMode
Syntax: Std_ReturnType Can_SetControllerMode(uint8 Controller, Can_StateTransitionType Transition)
Return value:
Std_ReturnType
E_OK: request accepted
E_NOT_OK: request not accepted, a development error occurred

~SWS_Can_00360 Can_CheckWakeup
Syntax: Std_ReturnType Can_CheckWakeup(uint8 Controller)
Return value:
Std_ReturnType
E_OK: API call has been accepted
E_NOT_OK: API call has not been accepted

~SWS_Can_00233 Can_Write
Syntax: Std_ReturnType Can_Write(Can_HwHandleType Hth, const Can_PduType* PduInfo)
Return value:
Std_ReturnType
E_OK: Write command has been accepted
E_NOT_OK: development error occurred
CAN_BUSY: No TX hardware buffer available or pre-emptive call of Can_Write that can't be implemented re-entrant (see Can_ReturnType)

Rename CAN_OK to E_OK and CAN_NOT_OK to E_NOT_OK:
~SWS_Can_00048
~SWS_Can_00089
7.11.5 Return Values
~SWS_Can_00198
~SWS_Can_00199

~SWS_Can_00200
~SWS_Can_00216
~SWS_Can_00217
~SWS_Can_00218
~SWS_CAN_00219
~SWS_CAN_00505
~SWS_CAN_00506
~SWS_Can_00212

=== CanIf ===
Adapt API Can_Write() to new signature:
* Figure 7.10 "Transmission request with multiple CAN Drivers - simplified"
* Figure 9.1 "Transmission request with a single CAN Driver"
* Figure 9.2 "Transmission request with multiple CAN Drivers"
* Figure 9.5 "Transmit confirmation with buffering"
* Figure 9.6 "Transmit Cancelation"
* Figure 9.7 "Trigger Transmit Request"

Adapt API Can_SetControllerMode() to new signature:
* Figure 9.11: Start CAN network
* Figure 9.13: BusOff recovery

Figure 9.13: Change typo "Cnange" to "Change"

~SWS_CANIF_00678: If all calls of Can_CheckWakeup() or CanTrcv_CheckWakeup() return E_NOT_OK to CanIf, then CanIf_CheckWakeup() shall return E_NOT_OK.

~SWS_CANIF_00720: If at least one function call of Can_CheckWakeup() or CanTrcv_CheckWakeup() returns E_OK to CanIf, then CanIf_CheckWakeup() shall return E_OK.

Rename CAN_OK to E_OK and CAN_NOT_OK to E_NOT_OK:
Note between SWS_CANIF_00162 and SWS_CANIF_00319
Table in chapter 9.7 Trigger Transmit Request
Table in chapter 9.11 Start CAN network

=== CanTrcv ===
Adapt API Can_SetControllerMode() to new signature:
* 9.3 De-Initialization (SPI Synchronous)
* 9.4 De-Initialization (SPI Asynchronous)

=== EcuSM ===

Adapt API Can_CheckWakeup() to new signature:
* Figure 42 CAN controller wake up by interrupt
* Figure 43 CAN controller or transceiver wake up by polling

=== TTCanIf ===
Adapt API Can_Write() to new signature:
* Figure 9.1: CAN Interface Time Triggered transmission with Job List

Correct API Can_TTReceive() which has return void instead of Can_ReturnType indeed:
* Figure 9.2: CAN Interface Time Triggered reception with Job List

Rename CAN_OK to E_OK and CAN_NOT_OK to E_NOT_OK:
~SWS_TtCanIf_00071

=== TTCanDrv ===
Rename CAN_OK to E_OK and CAN_NOT_OK to E_NOT_OK:
~SWS_TtCan_00014
~SWS_TtCan_00018
~SWS_TtCan_00022
~SWS_TtCan_00026
~SWS_TtCan_00059
~SWS_TtCan_00078
~SWS_TtCan_00112

=== XCP ===
Adapt API Can_Write() to new signature:
* Figure 5: Xcp on Can Transmit
–Last change on issue 77952 comment 22–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.24 Specification Item SWS_CAN_00219

Trace References:

none

Content:

If default development error detection for CanDrv is enabled: Can_Write() shall raise CAN_E_PARAM_POINTER and shall return CANE_NOT_OK if the parameter PduInfo is a null pointer.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #73570: No "default error" in AUTOSAR

  Problem description:

  The DET was renamed from development error tracer to default error tracer.

  This change was most of the time done automatically and unfortunately renamed "developement error" to "default error".

  "default error" should always be followed by "tracer", otherwise, "development error" is probably the right term.

  This could increase the impact (compared to my selection of impacted document, but formally, the configuration parameters *DevErrorDetect are not using the correct description:
  "Switches the Default Error Tracer (Det) detection and notification..."

  The parameter switches on/off the developement error detection. The DET does not need to be detected and can be present even when the parameter is set to false.

  Agreed solution:

  Rename "default error" to "development error" in all impacted documents, but not in an automated way (Do not change "default error tracer" to "developement error tracer"!)

  Blueprint/Example:
  - sub chapter is now called "7.x Default errors"
  - "[SWS_xxx_yyyyy]
  In case default error detection is enabled for the xxxx module: The xxxx module shall check API parameters for validity and report detected errors to the DET. ()"
  - "[SWS_xxx_yyyyy]
  If default error detection is enabled: the function shall check that the service xxx_Init was previously called. If the check fails, the function shall raise the default error XXX_E_NOT_INITIALIZED otherwise (if DET is disabled) return E_NOT_OK. ()"
  - "In case default errors are enabled,..."
  - "module raises the Default error XXX_E_TRANSITION"
  - "The DET provides services to store default errors"

...

The correct text would be:
- sub chapter is called "7.x Development errors"
- "[SWS_xxx_yyyyy]
In case development error detection is enabled for the xxxx module: The xxxx module shall check API parameters for validity and report detected development errors to the DET. ()"
- "[SWS_xxx_yyyyy]
If development error detection is enabled: the function shall check that the service xxx_Init was previously called. If the check fails, the function shall raise the development error XXX_E_NOT_INITIALIZED otherwise (if DET is disabled) return E_NOT_OK. ()"
- "In case development errors are enabled,..."
- "module raises the development error XXX_E_TRANSITION"
- "The DET provides services to store development errors"

Solution for SWS_RTE:
– SWS_RTE —
- Change 4.8 Default errors to 4.8 Development errors
- Change "Errors which can occur at runtime in the RTE are classified as default errors" to "Errors which can occur at runtime in the RTE are classified as development errors"
- Remove [SWS_Rte_07676]
- Change [SWS_RTE_06611]"If a violation is detected the RTE shall report a default error to the DET." to "If a violation is detected the RTE shall report a development error to the DET."
- Change [SWS_Rte_06631]
[SWS_Rte_06631] d The RTE shall use the OS Application Identifier as the Instance Id to enable the developer to identify in which runtime section of the RTE the error occurs. This Instance ID is even unique across multi cores and so implicitly allows the development error to be traced to a specific core. c(SRS_BSW_00337)

SRS_Libraries:
- In chapter "3 Acronyms and abbreviations": Rename "Development Error Tracer" to "Default Error Tracer"

SRS_SPALGeneral:
- In chapter "6.1.1.3.1 [SRS_SPAL_00157] ...": Rename "Development Error Tracer" to "Default Error Tracer"

- In chapter "6.1.1.4.2 [SRS_SPAL_12448] ...": Rename "Development Error Tracer" to "Default Error Tracer"

SRS_FlashTest:
- In chapter "6.1 Functional Requirements": Rename "Development Error Tracer" to "Default Error Tracer"
- In chapter "7 References":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_MFXLibrary:
- In chapter "2 Acronyms and abbreviations": Rename "Development Error Tracer" to "Default Error Tracer"

SWS_MemoryAbstractionInterface:
- In chapter "3.1 Input documents":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_FlexRayNetworkManagement:
- In chapter "3.3 Related AUTOSAR documents":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_CANStateManager:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_PDURouter:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_EEPROMDriver:

- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"
–Last change on issue 73570 comment 47–

BW-C-Level:

Application	Specification	Bus
1	1	1

- RfC #77952: [Can][CanIf] Incompatible return types of Can and CanTrcv

  Problem description:

  While CanTrcv only uses Std_ReturnType, Can uses Can_ReturnType in many places, even when only CAN_OK and CAN_NOT_OK are available.

  This leads to complicated code in CanIf, because it needs to implement separate checks for return values from CanTrcv and Can and cannot just combine the results.

  Agreed solution:

  === CanDrv ===

  Change of SWS_Can_00039 Can_ReturnType:
  * type change from enumeration to extra_litaral
  * Remove range element CAN_OK
  * Remove range element CAN_NOT_OK
  * Assign value "0x02" to range element "CAN_BUSY"
  * Description: Overlayed return value of Std_ReturnType for CAN driver API Can_Write().

  ~SWS_Can_00230 Can_SetControllerMode
  Syntax: Std_ReturnType Can_SetControllerMode(uint8 Controller, Can_StateTransitionType Transition)
  Return value:
  Std_ReturnType
  E_OK: request accepted
  E_NOT_OK: request not accepted, a development error occurred

  ~SWS_Can_00360 Can_CheckWakeup
  Syntax: Std_ReturnType Can_CheckWakeup(uint8 Controller)
  Return value:
  Std_ReturnType
  E_OK: API call has been accepted
  E_NOT_OK: API call has not been accepted

~SWS_Can_00233 Can_Write
Syntax: Std_ReturnType Can_Write(Can_HwHandleType Hth, const Can_PduType* PduInfo)
Return value:
Std_ReturnType
E_OK: Write command has been accepted
E_NOT_OK: development error occurred
CAN_BUSY: No TX hardware buffer available or pre-emptive call of Can_Write that can't be implemented re-entrant (see Can_ReturnType)

Rename CAN_OK to E_OK and CAN_NOT_OK to E_NOT_OK:
~SWS_Can_00048
~SWS_Can_00089
7.11.5 Return Values
~SWS_Can_00198
~SWS_Can_00199
~SWS_Can_00200
~SWS_Can_00216
~SWS_Can_00217
~SWS_Can_00218
~SWS_CAN_00219
~SWS_CAN_00505
~SWS_CAN_00506
~SWS_Can_00212

=== CanIf ===
Adapt API Can_Write() to new signature:
* Figure 7.10 "Transmission request with multiple CAN Drivers - simplified"
* Figure 9.1 "Transmission request with a single CAN Driver"
* Figure 9.2 "Transmission request with multiple CAN Drivers"
* Figure 9.5 "Transmit confirmation with buffering"
* Figure 9.6 "Transmit Cancelation"
* Figure 9.7 "Trigger Transmit Request"

Adapt API Can_SetControllerMode() to new signature:
* Figure 9.11: Start CAN network
* Figure 9.13: BusOff recovery

Figure 9.13: Change typo "Cnange" to "Change"

~SWS_CANIF_00678: If all calls of Can_CheckWakeup() or CanTrcv_CheckWakeup() return E_NOT_OK to CanIf, then CanIf_CheckWakeup()

shall return E_NOT_OK.

~SWS_CANIF_00720: If at least one function call of Can_CheckWakeup() or CanTrcv_CheckWakeup() returns E_OK to CanIf, then
CanIf_CheckWakeup() shall return E_OK.

Rename CAN_OK to E_OK and CAN_NOT_OK to E_NOT_OK:
Note between SWS_CANIF_00162 and SWS_CANIF_00319
Table in chapter 9.7 Trigger Transmit Request
Table in chapter 9.11 Start CAN network

=== CanTrcv ===
Adapt API Can_SetControllerMode() to new signature:
* 9.3 De-Initialization (SPI Synchronous)
* 9.4 De-Initialization (SPI Asynchronous)

=== EcuSM ===
Adapt API Can_CheckWakeup() to new signature:
* Figure 42 CAN controller wake up by interrupt
* Figure 43 CAN controller or transceiver wake up by polling

=== TTCanIf ===
Adapt API Can_Write() to new signature:
* Figure 9.1: CAN Interface Time Triggered transmission with Job List

Correct API Can_TTReceive() which has return void instead of Can_ReturnType indeed:
* Figure 9.2: CAN Interface Time Triggered reception with Job List

Rename CAN_OK to E_OK and CAN_NOT_OK to E_NOT_OK:
~SWS_TtCanIf_00071

=== TTCanDrv ===
Rename CAN_OK to E_OK and CAN_NOT_OK to E_NOT_OK:
~SWS_TtCan_00014
~SWS_TtCan_00018
~SWS_TtCan_00022
~SWS_TtCan_00026
~SWS_TtCan_00059
~SWS_TtCan_00078
~SWS_TtCan_00112

=== XCP ===

Adapt API Can_Write() to new signature:
* Figure 5: Xcp on Can Transmit
–Last change on issue 77952 comment 22–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.25 Specification Item SWS_Can_00222

Trace References:

none

Content:

Module	Imported Type
Can_GeneralTypes	Can_ControllerStateType
	Can_ErrorStateType
	Can_HwHandleType
	Can_HwType
	Can_IdType
	Can_PduType
	Can_ReturnType
Can_StateTransitionType	
ComStack_Types	IcomConfigIdType
	IcomSwitch_ErrorType
	PduIdType
	PduInfoType
EcuM	EcuM_WakeupSourceType
Icu	Icu_ChannelType
Os	CounterType
	StatusType
	TickRefType
Std_Types	Std_ReturnType
	Std_VersionInfoType

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77329: [Can] Removed type Can_StateTransitionType still used in Can_SetControllerMode function definition

Problem description:

With CP_R4.3.0 the type definition for Can_StateTransitionType has been removed from SWS_CANDriver.
In the function definition of Can_SetControllerMode (SWS_Can_00230) this type is still used in the signature syntax.

Within mentioned function calls (e.g. "Can_SetControllerMode(CAN_CS_STARTED)" in SWS_Can_00261) the enumeration values of type Can_ControllerStateType are used.

It shall be clarified if the type of the second function parameter in the syntax entry of SWS_Can_00230 shall be changed from Can_StateTransitionType to Can_ControllerStateType.

Agreed solution:

=== CanDrv ===
~SWS_Can_00230: Change Can_StateTransitionType to Can_ControllerStateType.
~SWS_Can_00222: Remove Can_StateTransitionType from Imported types

=== CanIf ===
~SWS_CANIF_00142: Remove Can_StateTransitionType from Can_GeneralTypes

Replace Can_StateTransitionType with Can_ControllerStateType of function call Can_SetControllerMode in chapters
- 9.11 (figure and table)
- 9.13 (figure)
–Last change on issue 77329 comment 12–

BW-C-Level:

Application	Specification	Bus
1	4	1

- RfC #77952: [Can][CanIf] Incompatible return types of Can and CanTrcv

Problem description:

While CanTrcv only uses Std_ReturnType, Can uses Can_ReturnType in many places, even when only CAN_OK and CAN_NOT_OK are available.

This leads to complicated code in CanIf, because it needs to implement separate checks for return values from CanTrcv and Can and cannot just combine the results.

Agreed solution:

=== CanDrv ===

Change of SWS_Can_00039 Can_ReturnType:
* type change from enumeration to extra_litaral
* Remove range element CAN_OK
* Remove range element CAN_NOT_OK
* Assign value "0x02" to range element "CAN_BUSY"
* Description: Overlayed return value of Std_ReturnType for CAN driver API Can_Write().

~SWS_Can_00230 Can_SetControllerMode
Syntax: Std_ReturnType Can_SetControllerMode(uint8 Controller, Can_StateTransitionType Transition)
Return value:
Std_ReturnType
E_OK: request accepted
E_NOT_OK: request not accepted, a development error occurred

~SWS_Can_00360 Can_CheckWakeup
Syntax: Std_ReturnType Can_CheckWakeup(uint8 Controller)
Return value:
Std_ReturnType
E_OK: API call has been accepted
E_NOT_OK: API call has not been accepted

~SWS_Can_00233 Can_Write
Syntax: Std_ReturnType Can_Write(Can_HwHandleType Hth, const Can_PduType* PduInfo)
Return value:
Std_ReturnType
E_OK: Write command has been accepted
E_NOT_OK: development error occurred
CAN_BUSY: No TX hardware buffer available or pre-emptive call of Can_Write that can't be implemented re-entrant (see Can_ReturnType)

Rename CAN_OK to E_OK and CAN_NOT_OK to E_NOT_OK:
~SWS_Can_00048
~SWS_Can_00089
7.11.5 Return Values
~SWS_Can_00198
~SWS_Can_00199
~SWS_Can_00200
~SWS_Can_00216

~SWS_Can_00217
~SWS_Can_00218
~SWS_CAN_00219
~SWS_CAN_00505
~SWS_CAN_00506
~SWS_Can_00212

=== CanIf ===
Adapt API Can_Write() to new signature:
* Figure 7.10 "Transmission request with multiple CAN Drivers - simplified"
* Figure 9.1 "Transmission request with a single CAN Driver"
* Figure 9.2 "Transmission request with multiple CAN Drivers"
* Figure 9.5 "Transmit confirmation with buffering"
* Figure 9.6 "Transmit Cancelation"
* Figure 9.7 "Trigger Transmit Request"

Adapt API Can_SetControllerMode() to new signature:
* Figure 9.11: Start CAN network
* Figure 9.13: BusOff recovery

Figure 9.13: Change typo "Cnange" to "Change"

~SWS_CANIF_00678: If all calls of Can_CheckWakeup() or CanTrcv_CheckWakeup() return E_NOT_OK to CanIf, then CanIf_CheckWakeup() shall return E_NOT_OK.

~SWS_CANIF_00720: If at least one function call of Can_CheckWakeup() or CanTrcv_CheckWakeup() returns E_OK to CanIf, then CanIf_CheckWakeup() shall return E_OK.

Rename CAN_OK to E_OK and CAN_NOT_OK to E_NOT_OK:
Note between SWS_CANIF_00162 and SWS_CANIF_00319
Table in chapter 9.7 Trigger Transmit Request
Table in chapter 9.11 Start CAN network

=== CanTrcv ===
Adapt API Can_SetControllerMode() to new signature:
* 9.3 De-Initialization (SPI Synchronous)
* 9.4 De-Initialization (SPI Asynchronous)

=== EcuSM ===
Adapt API Can_CheckWakeup() to new signature:
* Figure 42 CAN controller wake up by interrupt

* Figure 43 CAN controller or transceiver wake up by polling

=== TTCanIf ===
Adapt API Can_Write() to new signature:
* Figure 9.1: CAN Interface Time Triggered transmission with Job List

Correct API Can_TTReceive() which has return void instead of Can_ReturnType indeed:
* Figure 9.2: CAN Interface Time Triggered reception with Job List

Rename CAN_OK to E_OK and CAN_NOT_OK to E_NOT_OK:
~SWS_TtCanIf_00071

=== TTCanDrv ===
Rename CAN_OK to E_OK and CAN_NOT_OK to E_NOT_OK:
~SWS_TtCan_00014
~SWS_TtCan_00018
~SWS_TtCan_00022
~SWS_TtCan_00026
~SWS_TtCan_00059
~SWS_TtCan_00078
~SWS_TtCan_00112

=== XCP ===
Adapt API Can_Write() to new signature:
* Figure 5: Xcp on Can Transmit
–Last change on issue 77952 comment 22–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.26 Specification Item SWS_Can_00230

Trace References:

none

Content:

Service name:	Can_SetControllerModeCan_SetControllerMode

Syntax:	CanStd_ReturnType Can_SetControllerMode(uint8 Controller, Can_StateTransitionControllerStateType Transition)	
Service ID[hex]:	0x03	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	ControllerCan_SetController Mode.Controller	CAN controller for which the status shall be changed
	TransitionCan_SetController Mode.Transition	Transition value to request new CAN controller state
Parameters (inout):	None	
Parameters (out):	None	
Return value:	CanStd_ReturnType	CANE_OK: request accepted CANE_NOT_OK: request not accepted, a development error occurred
Description:	This function performs software triggered state transitions of the CAN controller State machine.	

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77329: [Can] Removed type Can_StateTransitionType still used in Can_SetControllerMode function definition

  Problem description:

  With CP_R4.3.0 the type definition for Can_StateTransitionType has been removed from SWS_CANDriver.
  In the function definition of Can_SetControllerMode (SWS_Can_00230) this type is still used in the signature syntax.

  Within mentioned function calls (e.g. "Can_SetControllerMode(CAN_CS_STARTED)" in SWS_Can_00261) the enumeration values of type Can_ControllerStateType are used.

  It shall be clarified if the type of the second function parameter in the syntax entry of SWS_Can_00230 shall be changed from Can_StateTransitionType to Can_ControllerStateType.

  Agreed solution:

  === CanDrv ===
  ~SWS_Can_00230: Change Can_StateTransitionType to Can_ControllerStateType.
  ~SWS_Can_00222: Remove Can_StateTransitionType from Imported types

=== CanIf ===
~SWS_CANIF_00142: Remove Can_StateTransitionType from Can_GeneralTypes

Replace Can_StateTransitionType with Can_ControllerStateType of function call Can_SetControllerMode in chapters
- 9.11 (figure and table)
- 9.13 (figure)
–Last change on issue 77329 comment 12–

BW-C-Level:

Application	Specification	Bus
1	4	1

- RfC #77952: [Can][CanIf] Incompatible return types of Can and CanTrcv

  Problem description:

  While CanTrcv only uses Std_ReturnType, Can uses Can_ReturnType in many places, even when only CAN_OK and CAN_NOT_OK are available.

  This leads to complicated code in CanIf, because it needs to implement separate checks for return values from CanTrcv and Can and cannot just combine the results.

  Agreed solution:

  === CanDrv ===

  Change of SWS_Can_00039 Can_ReturnType:
  * type change from enumeration to extra_litaral
  * Remove range element CAN_OK
  * Remove range element CAN_NOT_OK
  * Assign value "0x02" to range element "CAN_BUSY"
  * Description: Overlayed return value of Std_ReturnType for CAN driver API Can_Write().

  ~SWS_Can_00230 Can_SetControllerMode
  Syntax: Std_ReturnType Can_SetControllerMode(uint8 Controller, Can_StateTransitionType Transition)
  Return value:
  Std_ReturnType
  E_OK: request accepted
  E_NOT_OK: request not accepted, a development error occurred

  ~SWS_Can_00360 Can_CheckWakeup

Syntax: Std_ReturnType Can_CheckWakeup(uint8 Controller)
Return value:
Std_ReturnType
E_OK: API call has been accepted
E_NOT_OK: API call has not been accepted

~SWS_Can_00233 Can_Write
Syntax: Std_ReturnType Can_Write(Can_HwHandleType Hth, const Can_PduType* PduInfo)
Return value:
Std_ReturnType
E_OK: Write command has been accepted
E_NOT_OK: development error occurred
CAN_BUSY: No TX hardware buffer available or pre-emptive call of Can_Write that can't be implemented re-entrant (see Can_ReturnType)

Rename CAN_OK to E_OK and CAN_NOT_OK to E_NOT_OK:
~SWS_Can_00048
~SWS_Can_00089
7.11.5 Return Values
~SWS_Can_00198
~SWS_Can_00199
~SWS_Can_00200
~SWS_Can_00216
~SWS_Can_00217
~SWS_Can_00218
~SWS_CAN_00219
~SWS_CAN_00505
~SWS_CAN_00506
~SWS_Can_00212

=== CanIf ===
Adapt API Can_Write() to new signature:
* Figure 7.10 "Transmission request with multiple CAN Drivers - simplified"
* Figure 9.1 "Transmission request with a single CAN Driver"
* Figure 9.2 "Transmission request with multiple CAN Drivers"
* Figure 9.5 "Transmit confirmation with buffering"
* Figure 9.6 "Transmit Cancelation"
* Figure 9.7 "Trigger Transmit Request"

Adapt API Can_SetControllerMode() to new signature:
* Figure 9.11: Start CAN network
* Figure 9.13: BusOff recovery

Figure 9.13: Change typo "Cnange" to "Change"

~SWS_CANIF_00678: If all calls of Can_CheckWakeup() or CanTrcv_CheckWakeup() return E_NOT_OK to CanIf, then CanIf_CheckWakeup() shall return E_NOT_OK.

~SWS_CANIF_00720: If at least one function call of Can_CheckWakeup() or CanTrcv_CheckWakeup() returns E_OK to CanIf, then
CanIf_CheckWakeup() shall return E_OK.

Rename CAN_OK to E_OK and CAN_NOT_OK to E_NOT_OK:
Note between SWS_CANIF_00162 and SWS_CANIF_00319
Table in chapter 9.7 Trigger Transmit Request
Table in chapter 9.11 Start CAN network

=== CanTrcv ===
Adapt API Can_SetControllerMode() to new signature:
* 9.3 De-Initialization (SPI Synchronous)
* 9.4 De-Initialization (SPI Asynchronous)

=== EcuSM ===
Adapt API Can_CheckWakeup() to new signature:
* Figure 42 CAN controller wake up by interrupt
* Figure 43 CAN controller or transceiver wake up by polling

=== TTCanIf ===
Adapt API Can_Write() to new signature:
* Figure 9.1: CAN Interface Time Triggered transmission with Job List

Correct API Can_TTReceive() which has return void instead of Can_ReturnType indeed:
* Figure 9.2: CAN Interface Time Triggered reception with Job List

Rename CAN_OK to E_OK and CAN_NOT_OK to E_NOT_OK:
~SWS_TtCanIf_00071

=== TTCanDrv ===
Rename CAN_OK to E_OK and CAN_NOT_OK to E_NOT_OK:
~SWS_TtCan_00014
~SWS_TtCan_00018
~SWS_TtCan_00022
~SWS_TtCan_00026

~SWS_TtCan_00059
~SWS_TtCan_00078
~SWS_TtCan_00112

=== XCP ===
Adapt API Can_Write() to new signature:
* Figure 5: Xcp on Can Transmit
–Last change on issue 77952 comment 22–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.27 Specification Item SWS_Can_00233

Trace References:

SRS_BSW_00312

Content:

Service name:	Can_WriteCan_Write	
Syntax:	CanStd_ReturnType Can_Write(Can_HwHandleType Hth, const Can_PduType* PduInfo)	
Service ID[hex]:	0x06	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant (thread-safe)	
Parameters (in):	HthCan_Write.Hth	information which HW-transmit handle shall be used for transmit. Implicitly this is also the information about the controller to use because the Hth numbers are unique inside one hardware unit.
	PduInfoCan_Write.PduInfo	Pointer to SDU user memory, Data Length and Identifier.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	CanStd_ReturnType	CANE_OK: Write command has been accepted CANE_NOT_OK: development error occurred CAN_BUSY: No TX hardware buffer available or pre-emptive call of Can_Write that can't be implemented re-entrant (see Can_ReturnType)
Description:	This function is called by CanIf to pass a CAN message to CanDrv for transmission.	

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77952: [Can][CanIf] Incompatible return types of Can and CanTrcv

  Problem description:

  While CanTrcv only uses Std_ReturnType, Can uses Can_ReturnType in many places, even when only CAN_OK and CAN_NOT_OK are available.

  This leads to complicated code in CanIf, because it needs to implement separate checks for return values from CanTrcv and Can and cannot just combine the results.

  Agreed solution:

  === CanDrv ===

  Change of SWS_Can_00039 Can_ReturnType:
  * type change from enumeration to extra_litaral
  * Remove range element CAN_OK
  * Remove range element CAN_NOT_OK
  * Assign value "0x02" to range element "CAN_BUSY"
  * Description: Overlayed return value of Std_ReturnType for CAN driver API Can_Write().

  ~SWS_Can_00230 Can_SetControllerMode
  Syntax: Std_ReturnType Can_SetControllerMode(uint8 Controller, Can_StateTransitionType Transition)
  Return value:
  Std_ReturnType
  E_OK: request accepted
  E_NOT_OK: request not accepted, a development error occurred

  ~SWS_Can_00360 Can_CheckWakeup
  Syntax: Std_ReturnType Can_CheckWakeup(uint8 Controller)
  Return value:
  Std_ReturnType
  E_OK: API call has been accepted
  E_NOT_OK: API call has not been accepted

  ~SWS_Can_00233 Can_Write
  Syntax: Std_ReturnType Can_Write(Can_HwHandleType Hth, const Can_PduType* PduInfo)
  Return value:
  Std_ReturnType

E_OK: Write command has been accepted
E_NOT_OK: development error occurred
CAN_BUSY: No TX hardware buffer available or pre-emptive call of Can_Write that can't be implemented re-entrant (see Can_ReturnType)

Rename CAN_OK to E_OK and CAN_NOT_OK to E_NOT_OK:
~SWS_Can_00048
~SWS_Can_00089
7.11.5 Return Values
~SWS_Can_00198
~SWS_Can_00199
~SWS_Can_00200
~SWS_Can_00216
~SWS_Can_00217
~SWS_Can_00218
~SWS_CAN_00219
~SWS_CAN_00505
~SWS_CAN_00506
~SWS_Can_00212

=== CanIf ===
Adapt API Can_Write() to new signature:
* Figure 7.10 "Transmission request with multiple CAN Drivers - simplified"
* Figure 9.1 "Transmission request with a single CAN Driver"
* Figure 9.2 "Transmission request with multiple CAN Drivers"
* Figure 9.5 "Transmit confirmation with buffering"
* Figure 9.6 "Transmit Cancelation"
* Figure 9.7 "Trigger Transmit Request"

Adapt API Can_SetControllerMode() to new signature:
* Figure 9.11: Start CAN network
* Figure 9.13: BusOff recovery

Figure 9.13: Change typo "Cnange" to "Change"

~SWS_CANIF_00678: If all calls of Can_CheckWakeup() or CanTrcv_CheckWakeup() return E_NOT_OK to CanIf, then CanIf_CheckWakeup() shall return E_NOT_OK.

~SWS_CANIF_00720: If at least one function call of Can_CheckWakeup() or CanTrcv_CheckWakeup() returns E_OK to CanIf, then CanIf_CheckWakeup() shall return E_OK.

Rename CAN_OK to E_OK and CAN_NOT_OK to E_NOT_OK:
Note between SWS_CANIF_00162 and SWS_CANIF_00319
Table in chapter 9.7 Trigger Transmit Request
Table in chapter 9.11 Start CAN network

=== CanTrcv ===
Adapt API Can_SetControllerMode() to new signature:
* 9.3 De-Initialization (SPI Synchronous)
* 9.4 De-Initialization (SPI Asynchronous)

=== EcuSM ===
Adapt API Can_CheckWakeup() to new signature:
* Figure 42 CAN controller wake up by interrupt
* Figure 43 CAN controller or transceiver wake up by polling

=== TTCanIf ===
Adapt API Can_Write() to new signature:
* Figure 9.1: CAN Interface Time Triggered transmission with Job List

Correct API Can_TTReceive() which has return void instead of Can_ReturnType indeed:
* Figure 9.2: CAN Interface Time Triggered reception with Job List

Rename CAN_OK to E_OK and CAN_NOT_OK to E_NOT_OK:
~SWS_TtCanIf_00071

=== TTCanDrv ===
Rename CAN_OK to E_OK and CAN_NOT_OK to E_NOT_OK:
~SWS_TtCan_00014
~SWS_TtCan_00018
~SWS_TtCan_00022
~SWS_TtCan_00026
~SWS_TtCan_00059
~SWS_TtCan_00078
~SWS_TtCan_00112

=== XCP ===
Adapt API Can_Write() to new signature:
* Figure 5: Xcp on Can Transmit
–Last change on issue 77952 comment 22–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.28 Specification Item SWS_Can_00234

Trace References:

SRS_Can_01055

Content:

API function	Description
CanIf_ControllerBusOff	This service indicates a Controller BusOff event referring to the corresponding CAN Controller with the abstract CanIf ControllerId.
CanIf_ControllerModeIndication	This service indicates a controller state transition referring to the corresponding CAN controller with the abstract CanIf ControllerId.
CanIf_RxIndication	This service indicates a successful reception of a received CAN Rx L-PDU to the CanIf after passing all filters and validation checks.
CanIf_TxConfirmation	This service confirms a previously successfully processed transmission of a CAN TxPDU.
Det_ReportRuntimeError	Service to report runtime errors. If a callout has been configured then this callout shall be called.
GetCounterValue	This service reads the current count value of a counter (returning either the hardware timer ticks if counter is driven by hardware or the software ticks when user drives counter).

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

1.29 Specification Item SWS_Can_00360

Trace References:

none

Content:

Service name:	Can_CheckWakeupCan_CheckWakeup
Syntax:	CanStd_ReturnType Can_CheckWakeup(uint8 Controller)
Service ID[hex]:	0x0b
Sync/Async:	Synchronous

Reentrancy:	Non Reentrant	
Parameters (in):	ControllerCan_CheckWakeup.Controller	Controller to be checked for a wakeup.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	CanStd_ReturnType	CANE_OK: API call has been accepted CANE_NOT_OK: API call has not been accepted
Description:	This function checks if a wakeup has occurred for the given controller.	

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77952: [Can][CanIf] Incompatible return types of Can and CanTrcv

  Problem description:

  While CanTrcv only uses Std_ReturnType, Can uses Can_ReturnType in many places, even when only CAN_OK and CAN_NOT_OK are available.

  This leads to complicated code in CanIf, because it needs to implement separate checks for return values from CanTrcv and Can and cannot just combine the results.

  Agreed solution:

  === CanDrv ===

  Change of SWS_Can_00039 Can_ReturnType:
  * type change from enumeration to extra_litaral
  * Remove range element CAN_OK
  * Remove range element CAN_NOT_OK
  * Assign value "0x02" to range element "CAN_BUSY"
  * Description: Overlayed return value of Std_ReturnType for CAN driver API Can_Write().

  ~SWS_Can_00230 Can_SetControllerMode
  Syntax: Std_ReturnType Can_SetControllerMode(uint8 Controller, Can_StateTransitionType Transition)
  Return value:
  Std_ReturnType
  E_OK: request accepted
  E_NOT_OK: request not accepted, a development error occurred

  ~SWS_Can_00360 Can_CheckWakeup
  Syntax: Std_ReturnType Can_CheckWakeup(uint8 Controller)

Return value:
Std_ReturnType
E_OK: API call has been accepted
E_NOT_OK: API call has not been accepted

~SWS_Can_00233 Can_Write
Syntax: Std_ReturnType Can_Write(Can_HwHandleType Hth, const Can_PduType* PduInfo)
Return value:
Std_ReturnType
E_OK: Write command has been accepted
E_NOT_OK: development error occurred
CAN_BUSY: No TX hardware buffer available or pre-emptive call of Can_Write that can't be implemented re-entrant (see Can_ReturnType)

Rename CAN_OK to E_OK and CAN_NOT_OK to E_NOT_OK:
~SWS_Can_00048
~SWS_Can_00089
7.11.5 Return Values
~SWS_Can_00198
~SWS_Can_00199
~SWS_Can_00200
~SWS_Can_00216
~SWS_Can_00217
~SWS_Can_00218
~SWS_CAN_00219
~SWS_CAN_00505
~SWS_CAN_00506
~SWS_Can_00212

=== CanIf ===
Adapt API Can_Write() to new signature:
* Figure 7.10 "Transmission request with multiple CAN Drivers - simplified"
* Figure 9.1 "Transmission request with a single CAN Driver"
* Figure 9.2 "Transmission request with multiple CAN Drivers"
* Figure 9.5 "Transmit confirmation with buffering"
* Figure 9.6 "Transmit Cancelation"
* Figure 9.7 "Trigger Transmit Request"

Adapt API Can_SetControllerMode() to new signature:
* Figure 9.11: Start CAN network
* Figure 9.13: BusOff recovery

Figure 9.13: Change typo "Cnange" to "Change"

~SWS_CANIF_00678: If all calls of Can_CheckWakeup() or CanTrcv_CheckWakeup() return E_NOT_OK to CanIf, then CanIf_CheckWakeup() shall return E_NOT_OK.

~SWS_CANIF_00720: If at least one function call of Can_CheckWakeup() or CanTrcv_CheckWakeup() returns E_OK to CanIf, then
CanIf_CheckWakeup() shall return E_OK.

Rename CAN_OK to E_OK and CAN_NOT_OK to E_NOT_OK:
Note between SWS_CANIF_00162 and SWS_CANIF_00319
Table in chapter 9.7 Trigger Transmit Request
Table in chapter 9.11 Start CAN network

=== CanTrcv ===
Adapt API Can_SetControllerMode() to new signature:
* 9.3 De-Initialization (SPI Synchronous)
* 9.4 De-Initialization (SPI Asynchronous)

=== EcuSM ===
Adapt API Can_CheckWakeup() to new signature:
* Figure 42 CAN controller wake up by interrupt
* Figure 43 CAN controller or transceiver wake up by polling

=== TTCanIf ===
Adapt API Can_Write() to new signature:
* Figure 9.1: CAN Interface Time Triggered transmission with Job List

Correct API Can_TTReceive() which has return void instead of Can_ReturnType indeed:
* Figure 9.2: CAN Interface Time Triggered reception with Job List

Rename CAN_OK to E_OK and CAN_NOT_OK to E_NOT_OK:
~SWS_TtCanIf_00071

=== TTCanDrv ===
Rename CAN_OK to E_OK and CAN_NOT_OK to E_NOT_OK:
~SWS_TtCan_00014
~SWS_TtCan_00018
~SWS_TtCan_00022
~SWS_TtCan_00026
~SWS_TtCan_00059

~SWS_TtCan_00078
~SWS_TtCan_00112

=== XCP ===
Adapt API Can_Write() to new signature:
* Figure 5: Xcp on Can Transmit
–Last change on issue 77952 comment 22–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.30 Specification Item SWS_Can_00362

Trace References:

none

Content:

If default development error detection for the Can module is enabled: The function Can_CheckWakeup shall raise the error CAN_E_UNINIT if the driver is not yet initialized.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #73570: No "default error" in AUTOSAR

  Problem description:

  The DET was renamed from development error tracer to default error tracer.

  This change was most of the time done automatically and unfortunately renamed "developement error" to "default error".

  "default error" should always be followed by "tracer", otherwise, "development error" is probably the right term.

  This could increase the impact (compared to my selection of impacted document, but formally, the configuration parameters *DevErrorDetect are not using the correct description:
  "Switches the Default Error Tracer (Det) detection and notification..."

  The parameter switches on/off the developement error detection. The DET

does not need to be detected and can be present even when the parameter is set to false.

Agreed solution:

Rename "default error" to "development error" in all impacted documents, but not in an automated way (Do not change "default error tracer" to "developement error tracer"!)

Blueprint/Example:
- sub chapter is now called "7.x Default errors"
- "[SWS_xxx_yyyyy]
In case default error detection is enabled for the xxxx module: The xxxx module shall check API parameters for validity and report detected errors to the DET. ()"
- "[SWS_xxx_yyyyy]
If default error detection is enabled: the function shall check that the service xxx_Init was previously called. If the check fails, the function shall raise the default error XXX_E_NOT_INITIALIZED otherwise (if DET is disabled) return E_NOT_OK. ()"
- "In case default errors are enabled,..."
- "module raises the Default error XXX_E_TRANSITION"
- "The DET provides services to store default errors"
...

The correct text would be:
- sub chapter is called "7.x Development errors"
- "[SWS_xxx_yyyyy]
In case development error detection is enabled for the xxxx module: The xxxx module shall check API parameters for validity and report detected development errors to the DET. ()"
- "[SWS_xxx_yyyyy]
If development error detection is enabled: the function shall check that the service xxx_Init was previously called. If the check fails, the function shall raise the development error XXX_E_NOT_INITIALIZED otherwise (if DET is disabled) return E_NOT_OK. ()"
- "In case development errors are enabled,..."
- "module raises the development error XXX_E_TRANSITION"
- "The DET provides services to store development errors"

Solution for SWS_RTE:
– SWS_RTE —
- Change 4.8 Default errors to 4.8 Development errors
- Change "Errors which can occur at runtime in the RTE are classified as default errors" to "Errors which can occur at runtime in the RTE are classified as development

errors"
- Remove [SWS_Rte_07676]
- Change [SWS_RTE_06611]"If a violation is detected the RTE shall report a default error to the DET." to "If a violation is detected the RTE shall report a development error to the DET."
- Change [SWS_Rte_06631]
[SWS_Rte_06631] d The RTE shall use the OS Application Identifier as the Instance Id to enable the developer to identify in which runtime section of the RTE the error occurs. This Instance ID is even unique across multi cores and so implicitly allows the development error to be traced to a specific core. c(SRS_BSW_00337)

SRS_Libraries:
- In chapter "3 Acronyms and abbreviations": Rename "Development Error Tracer" to "Default Error Tracer"

SRS_SPALGeneral:
- In chapter "6.1.1.3.1 [SRS_SPAL_00157] ...": Rename "Development Error Tracer" to "Default Error Tracer"
- In chapter "6.1.1.4.2 [SRS_SPAL_12448] ...": Rename "Development Error Tracer" to "Default Error Tracer"

SRS_FlashTest:
- In chapter "6.1 Functional Requirements": Rename "Development Error Tracer" to "Default Error Tracer"
- In chapter "7 References":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_MFXLibrary:
- In chapter "2 Acronyms and abbreviations": Rename "Development Error Tracer" to "Default Error Tracer"

SWS_MemoryAbstractionInterface:
- In chapter "3.1 Input documents":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_FlexRayNetworkManagement:
- In chapter "3.3 Related AUTOSAR documents":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_CANStateManager:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_PDURouter:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_EEPROMDriver:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"
–Last change on issue 73570 comment 47–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.31 Specification Item SWS_Can_00363

Trace References:

none

Content:

If default development error detection for the Can module is enabled: The function Can_CheckWakeup shall raise the error CAN_E_PARAM_CONTROLLER if the parameter Controller is out of range.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #73570: No "default error" in AUTOSAR

Problem description:

The DET was renamed from development error tracer to default error tracer.

This change was most of the time done automatically and unfortunately renamed "developement error" to "default error".

"default error" should always be followed by "tracer", otherwise, "development error" is probably the right term.

This could increase the impact (compared to my selection of impacted document, but formally, the configuration parameters *DevErrorDetect are not using the correct description:
"Switches the Default Error Tracer (Det) detection and notification..."

The parameter switches on/off the developement error detection. The DET does not need to be detected and can be present even when the parameter is set to false.

Agreed solution:

Rename "default error" to "development error" in all impacted documents, but not in an automated way (Do not change "default error tracer" to "developement error tracer"!)

Blueprint/Example:
- sub chapter is now called "7.x Default errors"
- "[SWS_xxx_yyyyy]
In case default error detection is enabled for the xxxx module: The xxxx module shall check API parameters for validity and report detected errors to the DET. ()"
- "[SWS_xxx_yyyyy]
If default error detection is enabled: the function shall check that the service xxx_Init was previously called. If the check fails, the function shall raise the default error XXX_E_NOT_INITIALIZED otherwise (if DET is disabled) return E_NOT_OK. ()"
- "In case default errors are enabled,..."
- "module raises the Default error XXX_E_TRANSITION"
- "The DET provides services to store default errors"
...

The correct text would be:
- sub chapter is called "7.x Development errors"
- "[SWS_xxx_yyyyy]
In case development error detection is enabled for the xxxx module: The xxxx

module shall check API parameters for validity and report detected development errors to the DET. ()"
- "[SWS_xxx_yyyyy]
If development error detection is enabled: the function shall check that the service xxx_Init was previously called. If the check fails, the function shall raise the development error XXX_E_NOT_INITIALIZED otherwise (if DET is disabled) return E_NOT_OK. ()"
- "In case development errors are enabled,..."
- "module raises the development error XXX_E_TRANSITION"
- "The DET provides services to store development errors"

Solution for SWS_RTE:
– SWS_RTE —
- Change 4.8 Default errors to 4.8 Development errors
- Change "Errors which can occur at runtime in the RTE are classified as default errors" to "Errors which can occur at runtime in the RTE are classified as development errors"
- Remove [SWS_Rte_07676]
- Change [SWS_RTE_06611]"If a violation is detected the RTE shall report a default error to the DET." to "If a violation is detected the RTE shall report a development error to the DET."
- Change [SWS_Rte_06631]
[SWS_Rte_06631] d The RTE shall use the OS Application Identifier as the Instance Id to enable the developer to identify in which runtime section of the RTE the error occurs. This Instance ID is even unique across multi cores and so implicitly allows the development error to be traced to a specific core. c(SRS_BSW_00337)

SRS_Libraries:
- In chapter "3 Acronyms and abbreviations": Rename "Development Error Tracer" to "Default Error Tracer"

SRS_SPALGeneral:
- In chapter "6.1.1.3.1 [SRS_SPAL_00157] ...": Rename "Development Error Tracer" to "Default Error Tracer"
- In chapter "6.1.1.4.2 [SRS_SPAL_12448] ...": Rename "Development Error Tracer" to "Default Error Tracer"

SRS_FlashTest:
- In chapter "6.1 Functional Requirements": Rename "Development Error Tracer" to "Default Error Tracer"

- In chapter "7 References":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_MFXLibrary:
- In chapter "2 Acronyms and abbreviations": Rename "Development Error Tracer" to "Default Error Tracer"

SWS_MemoryAbstractionInterface:
- In chapter "3.1 Input documents":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_FlexRayNetworkManagement:
- In chapter "3.3 Related AUTOSAR documents":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_CANStateManager:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_PDURouter:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_EEPROMDriver:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"
–Last change on issue 73570 comment 47–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.32 Specification Item SWS_Can_00395

Trace References:

none

Content:

If the default error detection for the Can module is enabled, the Can module Can module shall raise the runtime error CAN_E_DATALOST in case of "overwrite" or "overrun" event detection.

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

- RfC #73570: No "default error" in AUTOSAR

  Problem description:

  The DET was renamed from development error tracer to default error tracer.

  This change was most of the time done automatically and unfortunately renamed "developement error" to "default error".

  "default error" should always be followed by "tracer", otherwise, "development error" is probably the right term.

  This could increase the impact (compared to my selection of impacted document, but formally, the configuration parameters *DevErrorDetect are not using the correct description:
  "Switches the Default Error Tracer (Det) detection and notification..."

  The parameter switches on/off the developement error detection. The DET does not need to be detected and can be present even when the parameter is set to false.

  Agreed solution:

  Rename "default error" to "development error" in all impacted documents, but not in an automated way (Do not change "default error tracer" to "developement error tracer"!)

  Blueprint/Example:
  - sub chapter is now called "7.x Default errors"
  - "[SWS_xxx_yyyyy]
  In case default error detection is enabled for the xxxx module: The xxxx module shall check API parameters for validity and report detected errors to the DET. ()"

- "[SWS_xxx_yyyyy]
If default error detection is enabled: the function shall check that the service xxx_Init was previously called. If the check fails, the function shall raise the default error XXX_E_NOT_INITIALIZED otherwise (if DET is disabled) return E_NOT_OK. ()"
- "In case default errors are enabled,..."
- "module raises the Default error XXX_E_TRANSITION"
- "The DET provides services to store default errors"
...

The correct text would be:
- sub chapter is called "7.x Development errors"
- "[SWS_xxx_yyyyy]
In case development error detection is enabled for the xxxx module: The xxxx module shall check API parameters for validity and report detected development errors to the DET. ()"
- "[SWS_xxx_yyyyy]
If development error detection is enabled: the function shall check that the service xxx_Init was previously called. If the check fails, the function shall raise the development error XXX_E_NOT_INITIALIZED otherwise (if DET is disabled) return E_NOT_OK. ()"
- "In case development errors are enabled,..."
- "module raises the development error XXX_E_TRANSITION"
- "The DET provides services to store development errors"

Solution for SWS_RTE:
– SWS_RTE —
- Change 4.8 Default errors to 4.8 Development errors
- Change "Errors which can occur at runtime in the RTE are classified as default errors" to "Errors which can occur at runtime in the RTE are classified as development errors"
- Remove [SWS_Rte_07676]
- Change [SWS_RTE_06611]"If a violation is detected the RTE shall report a default error to the DET." to "If a violation is detected the RTE shall report a development error to the DET."
- Change [SWS_Rte_06631]
[SWS_Rte_06631] d The RTE shall use the OS Application Identifier as the Instance Id to enable the developer to identify in which runtime section of the RTE the error occurs. This Instance ID is even unique across multi cores and so implicitly allows the development error to be traced to a specific core. c(SRS_BSW_00337)

SRS_Libraries:

- In chapter "3 Acronyms and abbreviations": Rename "Development Error Tracer" to "Default Error Tracer"

SRS_SPALGeneral:
- In chapter "6.1.1.3.1 [SRS_SPAL_00157] ...": Rename "Development Error Tracer" to "Default Error Tracer"
- In chapter "6.1.1.4.2 [SRS_SPAL_12448] ...": Rename "Development Error Tracer" to "Default Error Tracer"

SRS_FlashTest:
- In chapter "6.1 Functional Requirements": Rename "Development Error Tracer" to "Default Error Tracer"
- In chapter "7 References":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_MFXLibrary:
- In chapter "2 Acronyms and abbreviations": Rename "Development Error Tracer" to "Default Error Tracer"

SWS_MemoryAbstractionInterface:
- In chapter "3.1 Input documents":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_FlexRayNetworkManagement:
- In chapter "3.3 Related AUTOSAR documents":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_CANStateManager:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_PDURouter:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_EEPROMDriver:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"
–Last change on issue 73570 comment 47–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.33 Specification Item SWS_Can_00408

Trace References:

none

Content:

If default development error detection for the Can module is enabled: The function Can_Init shall raise the error CAN_E_TRANSITION if the CAN controllers are not in state UNINIT.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #73570: No "default error" in AUTOSAR

  Problem description:

  The DET was renamed from development error tracer to default error tracer.

  This change was most of the time done automatically and unfortunately renamed "developement error" to "default error".

  "default error" should always be followed by "tracer", otherwise, "development error" is probably the right term.

  This could increase the impact (compared to my selection of impacted document, but formally, the configuration parameters *DevErrorDetect are not using the correct description:

"Switches the Default Error Tracer (Det) detection and notification..."

The parameter switches on/off the developement error detection. The DET does not need to be detected and can be present even when the parameter is set to false.

Agreed solution:

Rename "default error" to "development error" in all impacted documents, but not in an automated way (Do not change "default error tracer" to "developement error tracer"!)

Blueprint/Example:
- sub chapter is now called "7.x Default errors"
- "[SWS_xxx_yyyyy]
In case default error detection is enabled for the xxxx module: The xxxx module shall check API parameters for validity and report detected errors to the DET. ()"
- "[SWS_xxx_yyyyy]
If default error detection is enabled: the function shall check that the service xxx_Init was previously called. If the check fails, the function shall raise the default error XXX_E_NOT_INITIALIZED otherwise (if DET is disabled) return E_NOT_OK. ()"
- "In case default errors are enabled,..."
- "module raises the Default error XXX_E_TRANSITION"
- "The DET provides services to store default errors"
...

The correct text would be:
- sub chapter is called "7.x Development errors"
- "[SWS_xxx_yyyyy]
In case development error detection is enabled for the xxxx module: The xxxx module shall check API parameters for validity and report detected development errors to the DET. ()"
- "[SWS_xxx_yyyyy]
If development error detection is enabled: the function shall check that the service xxx_Init was previously called. If the check fails, the function shall raise the development error XXX_E_NOT_INITIALIZED otherwise (if DET is disabled) return E_NOT_OK. ()"
- "In case development errors are enabled,..."
- "module raises the development error XXX_E_TRANSITION"
- "The DET provides services to store development errors"

Solution for SWS_RTE:
– SWS_RTE —

- Change 4.8 Default errors to 4.8 Development errors
- Change "Errors which can occur at runtime in the RTE are classified as default errors" to "Errors which can occur at runtime in the RTE are classified as development errors"
- Remove [SWS_Rte_07676]
- Change [SWS_RTE_06611]"If a violation is detected the RTE shall report a default error to the DET." to "If a violation is detected the RTE shall report a development error to the DET."
- Change [SWS_Rte_06631]
[SWS_Rte_06631] d The RTE shall use the OS Application Identifier as the Instance Id to enable the developer to identify in which runtime section of the RTE the error occurs. This Instance ID is even unique across multi cores and so implicitly allows the development error to be traced to a specific core. c(SRS_BSW_00337)

SRS_Libraries:
- In chapter "3 Acronyms and abbreviations": Rename "Development Error Tracer" to "Default Error Tracer"

SRS_SPALGeneral:
- In chapter "6.1.1.3.1 [SRS_SPAL_00157] ...": Rename "Development Error Tracer" to "Default Error Tracer"
- In chapter "6.1.1.4.2 [SRS_SPAL_12448] ...": Rename "Development Error Tracer" to "Default Error Tracer"

SRS_FlashTest:
- In chapter "6.1 Functional Requirements": Rename "Development Error Tracer" to "Default Error Tracer"
- In chapter "7 References":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_MFXLibrary:
- In chapter "2 Acronyms and abbreviations": Rename "Development Error Tracer" to "Default Error Tracer"

SWS_MemoryAbstractionInterface:
- In chapter "3.1 Input documents":

Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_FlexRayNetworkManagement:
- In chapter "3.3 Related AUTOSAR documents":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_CANStateManager:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_PDURouter:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_EEPROMDriver:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"
–Last change on issue 73570 comment 47–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.34 Specification Item SWS_Can_00416

Trace References:

none

Content:

Name:	Can_IdTypeCan_IdType
Type:	uint16, uint32

Range:	Standard32BitCan_Id Type.Standard32Bit	–	0..0x400007FF
	Standard16BitCan_Id Type.Standard16Bit	–	0..0x47FF
Extended32BitCan_Id Type.Extended32Bit	–	0..0xDFFFFFFF	
Description:	Represents the Identifier of an L-PDU. The two most significant bits specify the frame type: 00 CAN message with Standard CAN ID 01 CAN FD frame with Standard CAN ID 10 CAN message with Extended CAN ID 11 CAN FD frame with Extended CAN ID		

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #74108: [Can] [CanIf] Ambiguous handling of Can-FD flag on Can/CanIf interface

  Problem description:

  Clarification of Can-FD flag handling
  –Last change on issue 74108 comment 16–

  Agreed solution:

  ~SWS_Can_00416 Can_IdType

  - Remove uint16 from Type
  - Remove Standard16Bit from Range

  Editorial modify chapter "7.12 CAN FD Support", correct:

  "However, there may be cases where conventional CAN 2.0 messages need to betransmitted in networks supporting CAN-FD messages for example to facilitate CAN selective wakeup."

  to

  "However, there may be cases where conventional CAN 2.0 messages need to be transmitted in networks supporting CAN-FD messages for example to facilitate CAN selective wakeup."
  –Last change on issue 74108 comment 19–

  BW-C-Level:

Application	Specification	Bus
1	4	1

1.35 Specification Item SWS_CAN_00475

Trace References:

none

Content:

If default development error detection for CanDrv is enabled, then function Can_SetIcomConfiguration() shall report the default development error CAN_E_ICOM_CONFIG_INVALID if it is called with an invalid ConfigurationId (i.e. neither 0 nor any of the configured CanIcomConfigId).

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #73570: No "default error" in AUTOSAR

  Problem description:

  The DET was renamed from development error tracer to default error tracer.

  This change was most of the time done automatically and unfortunately renamed "developement error" to "default error".

  "default error" should always be followed by "tracer", otherwise, "development error" is probably the right term.

  This could increase the impact (compared to my selection of impacted document, but formally, the configuration parameters *DevErrorDetect are not using the correct description:
  "Switches the Default Error Tracer (Det) detection and notification..."

  The parameter switches on/off the developement error detection. The DET does not need to be detected and can be present even when the parameter is set to false.

  Agreed solution:

  Rename "default error" to "development error" in all impacted documents, but not in an automated way (Do not change "default error tracer" to "developement error tracer"!)

  Blueprint/Example:
  - sub chapter is now called "7.x Default errors"
  - "[SWS_xxx_yyyyy]
  In case default error detection is enabled for the xxxx module: The xxxx module

shall check API parameters for validity and report detected errors to the DET. ()"
- "[SWS_xxx_yyyyy]
If default error detection is enabled: the function shall check that the service xxx_Init was previously called. If the check fails, the function shall raise the default error XXX_E_NOT_INITIALIZED otherwise (if DET is disabled) return E_NOT_OK. ()"
- "In case default errors are enabled,..."
- "module raises the Default error XXX_E_TRANSITION"
- "The DET provides services to store default errors"
...

The correct text would be:
- sub chapter is called "7.x Development errors"
- "[SWS_xxx_yyyyy]
In case development error detection is enabled for the xxxx module: The xxxx module shall check API parameters for validity and report detected development errors to the DET. ()"
- "[SWS_xxx_yyyyy]
If development error detection is enabled: the function shall check that the service xxx_Init was previously called. If the check fails, the function shall raise the development error XXX_E_NOT_INITIALIZED otherwise (if DET is disabled) return E_NOT_OK. ()"
- "In case development errors are enabled,..."
- "module raises the development error XXX_E_TRANSITION"
- "The DET provides services to store development errors"

Solution for SWS_RTE:
– SWS_RTE —
- Change 4.8 Default errors to 4.8 Development errors
- Change "Errors which can occur at runtime in the RTE are classified as default errors" to "Errors which can occur at runtime in the RTE are classified as development errors"
- Remove [SWS_Rte_07676]
- Change [SWS_RTE_06611]"If a violation is detected the RTE shall report a default error to the DET." to "If a violation is detected the RTE shall report a development error to the DET."
- Change [SWS_Rte_06631]
[SWS_Rte_06631] d The RTE shall use the OS Application Identifier as the Instance Id to enable the developer to identify in which runtime section of the RTE the error occurs. This Instance ID is even unique across multi cores and so implicitly allows the development error to be traced to a specific core. c(SRS_BSW_00337)

SRS_Libraries:
- In chapter "3 Acronyms and abbreviations": Rename "Development Error Tracer" to "Default Error Tracer"

SRS_SPALGeneral:
- In chapter "6.1.1.3.1 [SRS_SPAL_00157] ...": Rename "Development Error Tracer" to "Default Error Tracer"
- In chapter "6.1.1.4.2 [SRS_SPAL_12448] ...": Rename "Development Error Tracer" to "Default Error Tracer"

SRS_FlashTest:
- In chapter "6.1 Functional Requirements": Rename "Development Error Tracer" to "Default Error Tracer"
- In chapter "7 References":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_MFXLibrary:
- In chapter "2 Acronyms and abbreviations": Rename "Development Error Tracer" to "Default Error Tracer"

SWS_MemoryAbstractionInterface:
- In chapter "3.1 Input documents":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_FlexRayNetworkManagement:
- In chapter "3.3 Related AUTOSAR documents":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_CANStateManager:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_PDURouter:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_EEPROMDriver:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"
–Last change on issue 73570 comment 47–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.36 Specification Item SWS_CAN_00492

Trace References:

none

Content:

If default development error detection for the Can module is enabled:

The function Can_SetBaudrate shall raise the error CAN_E_UNINIT and return

E_NOT_OK if the driver is not yet initialized.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #73570: No "default error" in AUTOSAR

  Problem description:

  The DET was renamed from development error tracer to default error tracer.

  This change was most of the time done automatically and unfortunately renamed "developement error" to "default error".

  "default error" should always be followed by "tracer", otherwise, "development error" is probably the right term.

  This could increase the impact (compared to my selection of impacted docu-

ment, but formally, the configuration parameters *DevErrorDetect are not using the correct description:
"Switches the Default Error Tracer (Det) detection and notification..."

The parameter switches on/off the developement error detection. The DET does not need to be detected and can be present even when the parameter is set to false.

Agreed solution:

Rename "default error" to "development error" in all impacted documents, but not in an automated way (Do not change "default error tracer" to "developement error tracer"!)

Blueprint/Example:
- sub chapter is now called "7.x Default errors"
- "[SWS_xxx_yyyyy]
In case default error detection is enabled for the xxxx module: The xxxx module shall check API parameters for validity and report detected errors to the DET. ()"
- "[SWS_xxx_yyyyy]
If default error detection is enabled: the function shall check that the service xxx_Init was previously called. If the check fails, the function shall raise the default error XXX_E_NOT_INITIALIZED otherwise (if DET is disabled) return E_NOT_OK. ()"
- "In case default errors are enabled,..."
- "module raises the Default error XXX_E_TRANSITION"
- "The DET provides services to store default errors"
...

The correct text would be:
- sub chapter is called "7.x Development errors"
- "[SWS_xxx_yyyyy]
In case development error detection is enabled for the xxxx module: The xxxx module shall check API parameters for validity and report detected development errors to the DET. ()"
- "[SWS_xxx_yyyyy]
If development error detection is enabled: the function shall check that the service xxx_Init was previously called. If the check fails, the function shall raise the development error XXX_E_NOT_INITIALIZED otherwise (if DET is disabled) return E_NOT_OK. ()"
- "In case development errors are enabled,..."
- "module raises the development error XXX_E_TRANSITION"
- "The DET provides services to store development errors"

Solution for SWS_RTE:
– SWS_RTE —
- Change 4.8 Default errors to 4.8 Development errors
- Change "Errors which can occur at runtime in the RTE are classified as default errors" to "Errors which can occur at runtime in the RTE are classified as development errors"
- Remove [SWS_Rte_07676]
- Change [SWS_RTE_06611]"If a violation is detected the RTE shall report a default error to the DET." to "If a violation is detected the RTE shall report a development error to the DET."
- Change [SWS_Rte_06631]
[SWS_Rte_06631] d The RTE shall use the OS Application Identifier as the Instance Id to enable the developer to identify in which runtime section of the RTE the error occurs. This Instance ID is even unique across multi cores and so implicitly allows the development error to be traced to a specific core. c(SRS_BSW_00337)

SRS_Libraries:
- In chapter "3 Acronyms and abbreviations": Rename "Development Error Tracer" to "Default Error Tracer"

SRS_SPALGeneral:
- In chapter "6.1.1.3.1 [SRS_SPAL_00157] ...": Rename "Development Error Tracer" to "Default Error Tracer"
- In chapter "6.1.1.4.2 [SRS_SPAL_12448] ...": Rename "Development Error Tracer" to "Default Error Tracer"

SRS_FlashTest:
- In chapter "6.1 Functional Requirements": Rename "Development Error Tracer" to "Default Error Tracer"
- In chapter "7 References":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_MFXLibrary:
- In chapter "2 Acronyms and abbreviations": Rename "Development Error Tracer" to "Default Error Tracer"

SWS_MemoryAbstractionInterface:
- In chapter "3.1 Input documents":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_FlexRayNetworkManagement:
- In chapter "3.3 Related AUTOSAR documents":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_CANStateManager:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_PDURouter:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_EEPROMDriver:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"
–Last change on issue 73570 comment 47–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.37 Specification Item SWS_CAN_00493

Trace References:

none

Content:

If default development error detection for the Can module is enabled:

The function Can_SetBaudrate shall raise the error CAN_E_PARAM_BAUDRATE and return E_NOT_OK if the parameter BaudRateConfigID has an invalid value.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #73570: No "default error" in AUTOSAR

  Problem description:

  The DET was renamed from development error tracer to default error tracer.

  This change was most of the time done automatically and unfortunately renamed "developement error" to "default error".

  "default error" should always be followed by "tracer", otherwise, "development error" is probably the right term.

  This could increase the impact (compared to my selection of impacted document, but formally, the configuration parameters *DevErrorDetect are not using the correct description:
  "Switches the Default Error Tracer (Det) detection and notification..."

  The parameter switches on/off the developement error detection. The DET does not need to be detected and can be present even when the parameter is set to false.

  Agreed solution:

  Rename "default error" to "development error" in all impacted documents, but not in an automated way (Do not change "default error tracer" to "developement error tracer"!)

  Blueprint/Example:
  - sub chapter is now called "7.x Default errors"
  - "[SWS_xxx_yyyyy]
  In case default error detection is enabled for the xxxx module: The xxxx module shall check API parameters for validity and report detected errors to the DET. ()"
  - "[SWS_xxx_yyyyy]
  If default error detection is enabled: the function shall check that the service xxx_Init was previously called. If the check fails, the function shall raise the default error XXX_E_NOT_INITIALIZED otherwise (if DET is disabled) return E_NOT_OK. ()"
  - "In case default errors are enabled,..."
  - "module raises the Default error XXX_E_TRANSITION"
  - "The DET provides services to store default errors"
  ...

The correct text would be:
- sub chapter is called "7.x Development errors"
- "[SWS_xxx_yyyyy]
In case development error detection is enabled for the xxxx module: The xxxx module shall check API parameters for validity and report detected development errors to the DET. ()"
- "[SWS_xxx_yyyyy]
If development error detection is enabled: the function shall check that the service xxx_Init was previously called. If the check fails, the function shall raise the development error XXX_E_NOT_INITIALIZED otherwise (if DET is disabled) return E_NOT_OK. ()"
- "In case development errors are enabled,..."
- "module raises the development error XXX_E_TRANSITION"
- "The DET provides services to store development errors"

Solution for SWS_RTE:
– SWS_RTE —
- Change 4.8 Default errors to 4.8 Development errors
- Change "Errors which can occur at runtime in the RTE are classified as default errors" to "Errors which can occur at runtime in the RTE are classified as development errors"
- Remove [SWS_Rte_07676]
- Change [SWS_RTE_06611]"If a violation is detected the RTE shall report a default error to the DET." to "If a violation is detected the RTE shall report a development error to the DET."
- Change [SWS_Rte_06631]
[SWS_Rte_06631] d The RTE shall use the OS Application Identifier as the Instance Id to enable the developer to identify in which runtime section of the RTE the error occurs. This Instance ID is even unique across multi cores and so implicitly allows the development error to be traced to a specific core. c(SRS_BSW_00337)

SRS_Libraries:
- In chapter "3 Acronyms and abbreviations": Rename "Development Error Tracer" to "Default Error Tracer"

SRS_SPALGeneral:
- In chapter "6.1.1.3.1 [SRS_SPAL_00157] ...": Rename "Development Error Tracer" to "Default Error Tracer"
- In chapter "6.1.1.4.2 [SRS_SPAL_12448] ...": Rename "Development Error Tracer"

to "Default Error Tracer"

SRS_FlashTest:
- In chapter "6.1 Functional Requirements": Rename "Development Error Tracer" to "Default Error Tracer"
- In chapter "7 References":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_MFXLibrary:
- In chapter "2 Acronyms and abbreviations": Rename "Development Error Tracer" to "Default Error Tracer"

SWS_MemoryAbstractionInterface:
- In chapter "3.1 Input documents":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_FlexRayNetworkManagement:
- In chapter "3.3 Related AUTOSAR documents":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_CANStateManager:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_PDURouter:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_EEPROMDriver:
- In chapter "3.1 Input documents": Rename "AU-

TOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"
–Last change on issue 73570 comment 47–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.38 Specification Item SWS_CAN_00494

Trace References:

none

Content:

If default development error detection for the Can module is enabled

the function Can_SetBaudrate shall raise the error CAN_E_PARAM_CONTROLLER and return E_NOT_OK if the parameter Controller is out of range.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #73570: No "default error" in AUTOSAR

  Problem description:

  The DET was renamed from development error tracer to default error tracer.

  This change was most of the time done automatically and unfortunately renamed "developement error" to "default error".

  "default error" should always be followed by "tracer", otherwise, "development error" is probably the right term.

  This could increase the impact (compared to my selection of impacted document, but formally, the configuration parameters *DevErrorDetect are not using the correct description:
  "Switches the Default Error Tracer (Det) detection and notification..."

  The parameter switches on/off the developement error detection. The DET does not need to be detected and can be present even when the parameter is set to false.

  Agreed solution:

Rename "default error" to "development error" in all impacted documents, but not in an automated way (Do not change "default error tracer" to "developement error tracer"!)

Blueprint/Example:
- sub chapter is now called "7.x Default errors"
- "[SWS_xxx_yyyyy]
In case default error detection is enabled for the xxxx module: The xxxx module shall check API parameters for validity and report detected errors to the DET. ()"
- "[SWS_xxx_yyyyy]
If default error detection is enabled: the function shall check that the service xxx_Init was previously called. If the check fails, the function shall raise the default error XXX_E_NOT_INITIALIZED otherwise (if DET is disabled) return E_NOT_OK. ()"
- "In case default errors are enabled,..."
- "module raises the Default error XXX_E_TRANSITION"
- "The DET provides services to store default errors"
...

The correct text would be:
- sub chapter is called "7.x Development errors"
- "[SWS_xxx_yyyyy]
In case development error detection is enabled for the xxxx module: The xxxx module shall check API parameters for validity and report detected development errors to the DET. ()"
- "[SWS_xxx_yyyyy]
If development error detection is enabled: the function shall check that the service xxx_Init was previously called. If the check fails, the function shall raise the development error XXX_E_NOT_INITIALIZED otherwise (if DET is disabled) return E_NOT_OK. ()"
- "In case development errors are enabled,..."
- "module raises the development error XXX_E_TRANSITION"
- "The DET provides services to store development errors"

Solution for SWS_RTE:
– SWS_RTE —
- Change 4.8 Default errors to 4.8 Development errors
- Change "Errors which can occur at runtime in the RTE are classified as default errors" to "Errors which can occur at runtime in the RTE are classified as development errors"
- Remove [SWS_Rte_07676]
- Change [SWS_RTE_06611]"If a violation is detected the RTE shall report a default error to the DET." to "If a violation is detected the RTE shall report a development

error to the DET."
- Change [SWS_Rte_06631]
[SWS_Rte_06631] d The RTE shall use the OS Application Identifier as the Instance Id to enable the developer to identify in which runtime section of the RTE the error occurs. This Instance ID is even unique across multi cores and so implicitly allows the development error to be traced to a specific core. c(SRS_BSW_00337)

SRS_Libraries:
- In chapter "3 Acronyms and abbreviations": Rename "Development Error Tracer" to "Default Error Tracer"

SRS_SPALGeneral:
- In chapter "6.1.1.3.1 [SRS_SPAL_00157] ...": Rename "Development Error Tracer" to "Default Error Tracer"
- In chapter "6.1.1.4.2 [SRS_SPAL_12448] ...": Rename "Development Error Tracer" to "Default Error Tracer"

SRS_FlashTest:
- In chapter "6.1 Functional Requirements": Rename "Development Error Tracer" to "Default Error Tracer"
- In chapter "7 References":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_MFXLibrary:
- In chapter "2 Acronyms and abbreviations": Rename "Development Error Tracer" to "Default Error Tracer"

SWS_MemoryAbstractionInterface:
- In chapter "3.1 Input documents":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_FlexRayNetworkManagement:
- In chapter "3.3 Related AUTOSAR documents":

Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_CANStateManager:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_PDURouter:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_EEPROMDriver:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"
–Last change on issue 73570 comment 47–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.39 Specification Item SWS_CAN_00504

Trace References:

none

Content:

If the trigger transmit API is enabled for the hardware object, Can_Write() shall interpret a null pointer as SDU (Can_PduType.Can_SduPtrType = NULL) as request for using the trigger transmit interface. If so and the hardware object is free, Can_Write() shall call CanIf_TriggerTransmit() with the maximum size of the message buffer to acquire the PDU's data.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #70123: [Can] TriggerTransmit does not work for dynamic length PDUs

  Problem description:

The CAN driver was inadvertently not addressed by RfC # 68517. The API description already implies the correct behavior, but it should be clearly stated, that TriggerTransmit needs to set the SduLength in the TriggerTransmit call to the size of the CAN hardware buffer.

Agreed solution:

Change SWS_CAN_00504 to:
/If the trigger transmit API is enabled for the hardware object, Can_Write() shall interpret a null pointer as SDU (Can_PduType.Can_SduPtrType = NULL) as request for using the trigger transmit interface. If so and the hardware object is free, Can_Write() shall call CanIf_TriggerTransmit() with the maximum size of the message buffer to acquire the PDUs data./

Add a note:
Using the message buffer size allows for late changes of the PDU size, e.g. if a container PDU receives another contained PDU between the call to Can_Write() and the call of CanIf_TriggerTransmit().
–Last change on issue 70123 comment 21–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.40 Specification Item SWS_CAN_00505

Trace References:

none

Content:

If default development error detection for CanDrv is enabled: Can_Write() shall raise CAN_E_PARAM_POINTER and shall return CANE_NOT_OK if the trigger transmit API is disabled for this hardware object (CanTriggerTransmitEnable = FALSE) and the SDU pointer inside PduInfo is a null pointer.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #73570: No "default error" in AUTOSAR

  Problem description:

  The DET was renamed from development error tracer to default error tracer.

This change was most of the time done automatically and unfortunately renamed "developement error" to "default error".

"default error" should always be followed by "tracer", otherwise, "development error" is probably the right term.

This could increase the impact (compared to my selection of impacted document, but formally, the configuration parameters *DevErrorDetect are not using the correct description:
"Switches the Default Error Tracer (Det) detection and notification..."

The parameter switches on/off the developement error detection. The DET does not need to be detected and can be present even when the parameter is set to false.

Agreed solution:

Rename "default error" to "development error" in all impacted documents, but not in an automated way (Do not change "default error tracer" to "developement error tracer"!)

Blueprint/Example:
- sub chapter is now called "7.x Default errors"
- "[SWS_xxx_yyyyy]
In case default error detection is enabled for the xxxx module: The xxxx module shall check API parameters for validity and report detected errors to the DET. ()"
- "[SWS_xxx_yyyyy]
If default error detection is enabled: the function shall check that the service xxx_Init was previously called. If the check fails, the function shall raise the default error XXX_E_NOT_INITIALIZED otherwise (if DET is disabled) return E_NOT_OK. ()"
- "In case default errors are enabled,..."
- "module raises the Default error XXX_E_TRANSITION"
- "The DET provides services to store default errors"
...

The correct text would be:
- sub chapter is called "7.x Development errors"
- "[SWS_xxx_yyyyy]
In case development error detection is enabled for the xxxx module: The xxxx module shall check API parameters for validity and report detected development errors to the DET. ()"
- "[SWS_xxx_yyyyy]
If development error detection is enabled: the function shall check that the service

xxx_Init was previously called. If the check fails, the function shall raise the development error XXX_E_NOT_INITIALIZED otherwise (if DET is disabled) return E_NOT_OK. ()"
- "In case development errors are enabled,..."
- "module raises the development error XXX_E_TRANSITION"
- "The DET provides services to store development errors"

Solution for SWS_RTE:
– SWS_RTE —
- Change 4.8 Default errors to 4.8 Development errors
- Change "Errors which can occur at runtime in the RTE are classified as default errors" to "Errors which can occur at runtime in the RTE are classified as development errors"
- Remove [SWS_Rte_07676]
- Change [SWS_RTE_06611]"If a violation is detected the RTE shall report a default error to the DET." to "If a violation is detected the RTE shall report a development error to the DET."
- Change [SWS_Rte_06631]
[SWS_Rte_06631] d The RTE shall use the OS Application Identifier as the Instance Id to enable the developer to identify in which runtime section of the RTE the error occurs. This Instance ID is even unique across multi cores and so implicitly allows the development error to be traced to a specific core. c(SRS_BSW_00337)

SRS_Libraries:
- In chapter "3 Acronyms and abbreviations": Rename "Development Error Tracer" to "Default Error Tracer"

SRS_SPALGeneral:
- In chapter "6.1.1.3.1 [SRS_SPAL_00157] ...": Rename "Development Error Tracer" to "Default Error Tracer"
- In chapter "6.1.1.4.2 [SRS_SPAL_12448] ...": Rename "Development Error Tracer" to "Default Error Tracer"

SRS_FlashTest:
- In chapter "6.1 Functional Requirements": Rename "Development Error Tracer" to "Default Error Tracer"
- In chapter "7 References":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_MFXLibrary:
- In chapter "2 Acronyms and abbreviations": Rename "Development Error Tracer" to "Default Error Tracer"

SWS_MemoryAbstractionInterface:
- In chapter "3.1 Input documents":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_FlexRayNetworkManagement:
- In chapter "3.3 Related AUTOSAR documents":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_CANStateManager:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_PDURouter:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_EEPROMDriver:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"
–Last change on issue 73570 comment 47–

BW-C-Level:

Application	Specification	Bus
1	1	1

- RfC #77952: [Can][CanIf] Incompatible return types of Can and CanTrcv

  Problem description:

While CanTrcv only uses Std_ReturnType, Can uses Can_ReturnType in many places, even when only CAN_OK and CAN_NOT_OK are available.

This leads to complicated code in CanIf, because it needs to implement separate checks for return values from CanTrcv and Can and cannot just combine the results.

Agreed solution:

=== CanDrv ===

Change of SWS_Can_00039 Can_ReturnType:
* type change from enumeration to extra_litaral
* Remove range element CAN_OK
* Remove range element CAN_NOT_OK
* Assign value "0x02" to range element "CAN_BUSY"
* Description: Overlayed return value of Std_ReturnType for CAN driver API Can_Write().

~SWS_Can_00230 Can_SetControllerMode
Syntax: Std_ReturnType Can_SetControllerMode(uint8 Controller, Can_StateTransitionType Transition)
Return value:
Std_ReturnType
E_OK: request accepted
E_NOT_OK: request not accepted, a development error occurred

~SWS_Can_00360 Can_CheckWakeup
Syntax: Std_ReturnType Can_CheckWakeup(uint8 Controller)
Return value:
Std_ReturnType
E_OK: API call has been accepted
E_NOT_OK: API call has not been accepted

~SWS_Can_00233 Can_Write
Syntax: Std_ReturnType Can_Write(Can_HwHandleType Hth, const Can_PduType* PduInfo)
Return value:
Std_ReturnType
E_OK: Write command has been accepted
E_NOT_OK: development error occurred
CAN_BUSY: No TX hardware buffer available or pre-emptive call of Can_Write that can't be implemented re-entrant (see Can_ReturnType)

Rename CAN_OK to E_OK and CAN_NOT_OK to E_NOT_OK:
~SWS_Can_00048
~SWS_Can_00089
7.11.5 Return Values
~SWS_Can_00198
~SWS_Can_00199
~SWS_Can_00200
~SWS_Can_00216
~SWS_Can_00217
~SWS_Can_00218
~SWS_CAN_00219
~SWS_CAN_00505
~SWS_CAN_00506
~SWS_Can_00212

=== CanIf ===
Adapt API Can_Write() to new signature:
* Figure 7.10 "Transmission request with multiple CAN Drivers - simplified"
* Figure 9.1 "Transmission request with a single CAN Driver"
* Figure 9.2 "Transmission request with multiple CAN Drivers"
* Figure 9.5 "Transmit confirmation with buffering"
* Figure 9.6 "Transmit Cancelation"
* Figure 9.7 "Trigger Transmit Request"

Adapt API Can_SetControllerMode() to new signature:
* Figure 9.11: Start CAN network
* Figure 9.13: BusOff recovery

Figure 9.13: Change typo "Cnange" to "Change"

~SWS_CANIF_00678: If all calls of Can_CheckWakeup() or CanTrcv_CheckWakeup() return E_NOT_OK to CanIf, then CanIf_CheckWakeup() shall return E_NOT_OK.

~SWS_CANIF_00720: If at least one function call of Can_CheckWakeup() or CanTrcv_CheckWakeup() returns E_OK to CanIf, then
CanIf_CheckWakeup() shall return E_OK.

Rename CAN_OK to E_OK and CAN_NOT_OK to E_NOT_OK:
Note between SWS_CANIF_00162 and SWS_CANIF_00319
Table in chapter 9.7 Trigger Transmit Request
Table in chapter 9.11 Start CAN network

=== CanTrcv ===
Adapt API Can_SetControllerMode() to new signature:
* 9.3 De-Initialization (SPI Synchronous)
* 9.4 De-Initialization (SPI Asynchronous)

=== EcuSM ===
Adapt API Can_CheckWakeup() to new signature:
* Figure 42 CAN controller wake up by interrupt
* Figure 43 CAN controller or transceiver wake up by polling

=== TTCanIf ===
Adapt API Can_Write() to new signature:
* Figure 9.1: CAN Interface Time Triggered transmission with Job List

Correct API Can_TTReceive() which has return void instead of Can_ReturnType indeed:
* Figure 9.2: CAN Interface Time Triggered reception with Job List

Rename CAN_OK to E_OK and CAN_NOT_OK to E_NOT_OK:
~SWS_TtCanIf_00071

=== TTCanDrv ===
Rename CAN_OK to E_OK and CAN_NOT_OK to E_NOT_OK:
~SWS_TtCan_00014
~SWS_TtCan_00018
~SWS_TtCan_00022
~SWS_TtCan_00026
~SWS_TtCan_00059
~SWS_TtCan_00078
~SWS_TtCan_00112

=== XCP ===
Adapt API Can_Write() to new signature:
* Figure 5: Xcp on Can Transmit
–Last change on issue 77952 comment 22–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.41 Specification Item SWS_CAN_00506

Trace References:

SRS_BSW_00449, SRS_BSW_00357, SRS_BSW_00369, SRS_Can_01130

Content:

Can_Write() shall return CANE_NOT_OK if the trigger transmit API (CanIf_TriggerTransmit()) returns E_NOT_OK.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77952: [Can][CanIf] Incompatible return types of Can and CanTrcv

  Problem description:

  While CanTrcv only uses Std_ReturnType, Can uses Can_ReturnType in many places, even when only CAN_OK and CAN_NOT_OK are available.

  This leads to complicated code in CanIf, because it needs to implement separate checks for return values from CanTrcv and Can and cannot just combine the results.

  Agreed solution:

  === CanDrv ===

  Change of SWS_Can_00039 Can_ReturnType:
  * type change from enumeration to extra_litaral
  * Remove range element CAN_OK
  * Remove range element CAN_NOT_OK
  * Assign value "0x02" to range element "CAN_BUSY"
  * Description: Overlayed return value of Std_ReturnType for CAN driver API Can_Write().

  ~SWS_Can_00230 Can_SetControllerMode
  Syntax: Std_ReturnType Can_SetControllerMode(uint8 Controller, Can_StateTransitionType Transition)
  Return value:
  Std_ReturnType
  E_OK: request accepted
  E_NOT_OK: request not accepted, a development error occurred

  ~SWS_Can_00360 Can_CheckWakeup
  Syntax: Std_ReturnType Can_CheckWakeup(uint8 Controller)

Return value:
Std_ReturnType
E_OK: API call has been accepted
E_NOT_OK: API call has not been accepted

~SWS_Can_00233 Can_Write
Syntax: Std_ReturnType Can_Write(Can_HwHandleType Hth, const Can_PduType* PduInfo)
Return value:
Std_ReturnType
E_OK: Write command has been accepted
E_NOT_OK: development error occurred
CAN_BUSY: No TX hardware buffer available or pre-emptive call of Can_Write that can't be implemented re-entrant (see Can_ReturnType)

Rename CAN_OK to E_OK and CAN_NOT_OK to E_NOT_OK:
~SWS_Can_00048
~SWS_Can_00089
7.11.5 Return Values
~SWS_Can_00198
~SWS_Can_00199
~SWS_Can_00200
~SWS_Can_00216
~SWS_Can_00217
~SWS_Can_00218
~SWS_CAN_00219
~SWS_CAN_00505
~SWS_CAN_00506
~SWS_Can_00212

=== CanIf ===
Adapt API Can_Write() to new signature:
* Figure 7.10 "Transmission request with multiple CAN Drivers - simplified"
* Figure 9.1 "Transmission request with a single CAN Driver"
* Figure 9.2 "Transmission request with multiple CAN Drivers"
* Figure 9.5 "Transmit confirmation with buffering"
* Figure 9.6 "Transmit Cancelation"
* Figure 9.7 "Trigger Transmit Request"

Adapt API Can_SetControllerMode() to new signature:
* Figure 9.11: Start CAN network
* Figure 9.13: BusOff recovery

Figure 9.13: Change typo "Cnange" to "Change"

~SWS_CANIF_00678: If all calls of Can_CheckWakeup() or CanTrcv_CheckWakeup() return E_NOT_OK to CanIf, then CanIf_CheckWakeup() shall return E_NOT_OK.

~SWS_CANIF_00720: If at least one function call of Can_CheckWakeup() or CanTrcv_CheckWakeup() returns E_OK to CanIf, then
CanIf_CheckWakeup() shall return E_OK.

Rename CAN_OK to E_OK and CAN_NOT_OK to E_NOT_OK:
Note between SWS_CANIF_00162 and SWS_CANIF_00319
Table in chapter 9.7 Trigger Transmit Request
Table in chapter 9.11 Start CAN network

=== CanTrcv ===
Adapt API Can_SetControllerMode() to new signature:
* 9.3 De-Initialization (SPI Synchronous)
* 9.4 De-Initialization (SPI Asynchronous)

=== EcuSM ===
Adapt API Can_CheckWakeup() to new signature:
* Figure 42 CAN controller wake up by interrupt
* Figure 43 CAN controller or transceiver wake up by polling

=== TTCanIf ===
Adapt API Can_Write() to new signature:
* Figure 9.1: CAN Interface Time Triggered transmission with Job List

Correct API Can_TTReceive() which has return void instead of Can_ReturnType indeed:
* Figure 9.2: CAN Interface Time Triggered reception with Job List

Rename CAN_OK to E_OK and CAN_NOT_OK to E_NOT_OK:
~SWS_TtCanIf_00071

=== TTCanDrv ===
Rename CAN_OK to E_OK and CAN_NOT_OK to E_NOT_OK:
~SWS_TtCan_00014
~SWS_TtCan_00018
~SWS_TtCan_00022
~SWS_TtCan_00026
~SWS_TtCan_00059

~SWS_TtCan_00078
~SWS_TtCan_00112

=== XCP ===
Adapt API Can_Write() to new signature:
* Figure 5: Xcp on Can Transmit
–Last change on issue 77952 comment 22–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.42 Specification Item SWS_Can_91005

Trace References:

SRS_BSW_00406, SRS_BSW_00416

Content:

If default development error detection for the Can module is enabled: if the module is not yet initialized, the function Can_GetControllerErrorState shall raise default development error CAN_E_UNINIT and return E_NOT_OK.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #73570: No "default error" in AUTOSAR

  Problem description:

  The DET was renamed from development error tracer to default error tracer.

  This change was most of the time done automatically and unfortunately renamed "developement error" to "default error".

  "default error" should always be followed by "tracer", otherwise, "development error" is probably the right term.

  This could increase the impact (compared to my selection of impacted document, but formally, the configuration parameters *DevErrorDetect are not using the correct description:
  "Switches the Default Error Tracer (Det) detection and notification..."

  The parameter switches on/off the developement error detection. The DET

does not need to be detected and can be present even when the parameter is set to false.

Agreed solution:

Rename "default error" to "development error" in all impacted documents, but not in an automated way (Do not change "default error tracer" to "developement error tracer"!)

Blueprint/Example:
- sub chapter is now called "7.x Default errors"
- "[SWS_xxx_yyyyy]
In case default error detection is enabled for the xxxx module: The xxxx module shall check API parameters for validity and report detected errors to the DET. ()"
- "[SWS_xxx_yyyyy]
If default error detection is enabled: the function shall check that the service xxx_Init was previously called. If the check fails, the function shall raise the default error XXX_E_NOT_INITIALIZED otherwise (if DET is disabled) return E_NOT_OK. ()"
- "In case default errors are enabled,..."
- "module raises the Default error XXX_E_TRANSITION"
- "The DET provides services to store default errors"
...

The correct text would be:
- sub chapter is called "7.x Development errors"
- "[SWS_xxx_yyyyy]
In case development error detection is enabled for the xxxx module: The xxxx module shall check API parameters for validity and report detected development errors to the DET. ()"
- "[SWS_xxx_yyyyy]
If development error detection is enabled: the function shall check that the service xxx_Init was previously called. If the check fails, the function shall raise the development error XXX_E_NOT_INITIALIZED otherwise (if DET is disabled) return E_NOT_OK. ()"
- "In case development errors are enabled,..."
- "module raises the development error XXX_E_TRANSITION"
- "The DET provides services to store development errors"

Solution for SWS_RTE:
– SWS_RTE —
- Change 4.8 Default errors to 4.8 Development errors
- Change "Errors which can occur at runtime in the RTE are classified as default errors" to "Errors which can occur at runtime in the RTE are classified as development

errors"
- Remove [SWS_Rte_07676]
- Change [SWS_RTE_06611]"If a violation is detected the RTE shall report a default error to the DET." to "If a violation is detected the RTE shall report a development error to the DET."
- Change [SWS_Rte_06631]
[SWS_Rte_06631] d The RTE shall use the OS Application Identifier as the Instance Id to enable the developer to identify in which runtime section of the RTE the error occurs. This Instance ID is even unique across multi cores and so implicitly allows the development error to be traced to a specific core. c(SRS_BSW_00337)

SRS_Libraries:
- In chapter "3 Acronyms and abbreviations": Rename "Development Error Tracer" to "Default Error Tracer"

SRS_SPALGeneral:
- In chapter "6.1.1.3.1 [SRS_SPAL_00157] ...": Rename "Development Error Tracer" to "Default Error Tracer"
- In chapter "6.1.1.4.2 [SRS_SPAL_12448] ...": Rename "Development Error Tracer" to "Default Error Tracer"

SRS_FlashTest:
- In chapter "6.1 Functional Requirements": Rename "Development Error Tracer" to "Default Error Tracer"
- In chapter "7 References":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_MFXLibrary:
- In chapter "2 Acronyms and abbreviations": Rename "Development Error Tracer" to "Default Error Tracer"

SWS_MemoryAbstractionInterface:
- In chapter "3.1 Input documents":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_FlexRayNetworkManagement:
- In chapter "3.3 Related AUTOSAR documents":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_CANStateManager:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_PDURouter:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_EEPROMDriver:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"
–Last change on issue 73570 comment 47–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.43 Specification Item SWS_Can_91006

Trace References:

SRS_BSW_00323

Content:

If default development error detection for the Can module is enabled: if the parameter ControllerId is out of range, the function Can_GetControllerErrorState shall raise default development error CAN_E_PARAM_CONTROLLER and return E_NOT_OK.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #73570: No "default error" in AUTOSAR

Problem description:

The DET was renamed from development error tracer to default error tracer.

This change was most of the time done automatically and unfortunately renamed "developement error" to "default error".

"default error" should always be followed by "tracer", otherwise, "development error" is probably the right term.

This could increase the impact (compared to my selection of impacted document, but formally, the configuration parameters *DevErrorDetect are not using the correct description:
"Switches the Default Error Tracer (Det) detection and notification..."

The parameter switches on/off the developement error detection. The DET does not need to be detected and can be present even when the parameter is set to false.

Agreed solution:

Rename "default error" to "development error" in all impacted documents, but not in an automated way (Do not change "default error tracer" to "developement error tracer"!)

Blueprint/Example:
- sub chapter is now called "7.x Default errors"
- "[SWS_xxx_yyyyy]
In case default error detection is enabled for the xxxx module: The xxxx module shall check API parameters for validity and report detected errors to the DET. ()"
- "[SWS_xxx_yyyyy]
If default error detection is enabled: the function shall check that the service xxx_Init was previously called. If the check fails, the function shall raise the default error XXX_E_NOT_INITIALIZED otherwise (if DET is disabled) return E_NOT_OK. ()"
- "In case default errors are enabled,..."
- "module raises the Default error XXX_E_TRANSITION"
- "The DET provides services to store default errors"
...

The correct text would be:
- sub chapter is called "7.x Development errors"
- "[SWS_xxx_yyyyy]
In case development error detection is enabled for the xxxx module: The xxxx

module shall check API parameters for validity and report detected development errors to the DET. ()"
- "[SWS_xxx_yyyyy]
If development error detection is enabled: the function shall check that the service xxx_Init was previously called. If the check fails, the function shall raise the development error XXX_E_NOT_INITIALIZED otherwise (if DET is disabled) return E_NOT_OK. ()"
- "In case development errors are enabled,..."
- "module raises the development error XXX_E_TRANSITION"
- "The DET provides services to store development errors"

Solution for SWS_RTE:
– SWS_RTE —
- Change 4.8 Default errors to 4.8 Development errors
- Change "Errors which can occur at runtime in the RTE are classified as default errors" to "Errors which can occur at runtime in the RTE are classified as development errors"
- Remove [SWS_Rte_07676]
- Change [SWS_RTE_06611]"If a violation is detected the RTE shall report a default error to the DET." to "If a violation is detected the RTE shall report a development error to the DET."
- Change [SWS_Rte_06631]
[SWS_Rte_06631] d The RTE shall use the OS Application Identifier as the Instance Id to enable the developer to identify in which runtime section of the RTE the error occurs. This Instance ID is even unique across multi cores and so implicitly allows the development error to be traced to a specific core. c(SRS_BSW_00337)

SRS_Libraries:
- In chapter "3 Acronyms and abbreviations": Rename "Development Error Tracer" to "Default Error Tracer"

SRS_SPALGeneral:
- In chapter "6.1.1.3.1 [SRS_SPAL_00157] ...": Rename "Development Error Tracer" to "Default Error Tracer"
- In chapter "6.1.1.4.2 [SRS_SPAL_12448] ...": Rename "Development Error Tracer" to "Default Error Tracer"

SRS_FlashTest:
- In chapter "6.1 Functional Requirements": Rename "Development Error Tracer" to "Default Error Tracer"

- In chapter "7 References":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_MFXLibrary:
- In chapter "2 Acronyms and abbreviations": Rename "Development Error Tracer" to "Default Error Tracer"

SWS_MemoryAbstractionInterface:
- In chapter "3.1 Input documents":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_FlexRayNetworkManagement:
- In chapter "3.3 Related AUTOSAR documents":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_CANStateManager:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_PDURouter:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_EEPROMDriver:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"
–Last change on issue 73570 comment 47–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.44 Specification Item SWS_Can_91007

Trace References:

SRS_BSW_00323

Content:

If default development error detection for the Can module is enabled: if the parameter ErrorStatePtr is a null pointer, the function Can_GetControllerErrorState shall raise default development error CAN_E_PARAM_POINTER and return E_NOT_OK.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #73570: No "default error" in AUTOSAR

  Problem description:

  The DET was renamed from development error tracer to default error tracer.

  This change was most of the time done automatically and unfortunately renamed "developement error" to "default error".

  "default error" should always be followed by "tracer", otherwise, "development error" is probably the right term.

  This could increase the impact (compared to my selection of impacted document, but formally, the configuration parameters *DevErrorDetect are not using the correct description:
  "Switches the Default Error Tracer (Det) detection and notification..."

  The parameter switches on/off the developement error detection. The DET does not need to be detected and can be present even when the parameter is set to false.

  Agreed solution:

  Rename "default error" to "development error" in all impacted documents, but not in an automated way (Do not change "default error tracer" to "developement error tracer"!)

  Blueprint/Example:
  - sub chapter is now called "7.x Default errors"
  - "[SWS_xxx_yyyyy]
  In case default error detection is enabled for the xxxx module: The xxxx module shall check API parameters for validity and report detected errors to the DET. ()"

- "[SWS_xxx_yyyyy]
If default error detection is enabled: the function shall check that the service xxx_Init was previously called. If the check fails, the function shall raise the default error XXX_E_NOT_INITIALIZED otherwise (if DET is disabled) return E_NOT_OK. ()"
- "In case default errors are enabled,..."
- "module raises the Default error XXX_E_TRANSITION"
- "The DET provides services to store default errors"
...

The correct text would be:
- sub chapter is called "7.x Development errors"
- "[SWS_xxx_yyyyy]
In case development error detection is enabled for the xxxx module: The xxxx module shall check API parameters for validity and report detected development errors to the DET. ()"
- "[SWS_xxx_yyyyy]
If development error detection is enabled: the function shall check that the service xxx_Init was previously called. If the check fails, the function shall raise the development error XXX_E_NOT_INITIALIZED otherwise (if DET is disabled) return E_NOT_OK. ()"
- "In case development errors are enabled,..."
- "module raises the development error XXX_E_TRANSITION"
- "The DET provides services to store development errors"

Solution for SWS_RTE:
– SWS_RTE —
- Change 4.8 Default errors to 4.8 Development errors
- Change "Errors which can occur at runtime in the RTE are classified as default errors" to "Errors which can occur at runtime in the RTE are classified as development errors"
- Remove [SWS_Rte_07676]
- Change [SWS_RTE_06611]"If a violation is detected the RTE shall report a default error to the DET." to "If a violation is detected the RTE shall report a development error to the DET."
- Change [SWS_Rte_06631]
[SWS_Rte_06631] d The RTE shall use the OS Application Identifier as the Instance Id to enable the developer to identify in which runtime section of the RTE the error occurs. This Instance ID is even unique across multi cores and so implicitly allows the development error to be traced to a specific core. c(SRS_BSW_00337)

SRS_Libraries:

- In chapter "3 Acronyms and abbreviations": Rename "Development Error Tracer" to "Default Error Tracer"

SRS_SPALGeneral:
- In chapter "6.1.1.3.1 [SRS_SPAL_00157] ...": Rename "Development Error Tracer" to "Default Error Tracer"
- In chapter "6.1.1.4.2 [SRS_SPAL_12448] ...": Rename "Development Error Tracer" to "Default Error Tracer"

SRS_FlashTest:
- In chapter "6.1 Functional Requirements": Rename "Development Error Tracer" to "Default Error Tracer"
- In chapter "7 References":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_MFXLibrary:
- In chapter "2 Acronyms and abbreviations": Rename "Development Error Tracer" to "Default Error Tracer"

SWS_MemoryAbstractionInterface:
- In chapter "3.1 Input documents":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_FlexRayNetworkManagement:
- In chapter "3.3 Related AUTOSAR documents":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_CANStateManager:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_PDURouter:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_EEPROMDriver:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"
–Last change on issue 73570 comment 47–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.45 Specification Item SWS_Can_91016

Trace References:

SRS_BSW_00406, SRS_BSW_00416

Content:

If default development error detection for the Can module is enabled:

The function Can_GetControllerMode shall raise the error CAN_E_UNINIT and return

E_NOT_OK if the driver is not yet initialized.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #73570: No "default error" in AUTOSAR

  Problem description:

  The DET was renamed from development error tracer to default error tracer.

  This change was most of the time done automatically and unfortunately renamed "developement error" to "default error".

  "default error" should always be followed by "tracer", otherwise, "development error" is probably the right term.

  This could increase the impact (compared to my selection of impacted document, but formally, the configuration parameters *DevErrorDetect are not using the

correct description:
"Switches the Default Error Tracer (Det) detection and notification..."

The parameter switches on/off the developement error detection. The DET does not need to be detected and can be present even when the parameter is set to false.

Agreed solution:

Rename "default error" to "development error" in all impacted documents, but not in an automated way (Do not change "default error tracer" to "developement error tracer"!)

Blueprint/Example:
- sub chapter is now called "7.x Default errors"
- "[SWS_xxx_yyyyy]
In case default error detection is enabled for the xxxx module: The xxxx module shall check API parameters for validity and report detected errors to the DET. ()"
- "[SWS_xxx_yyyyy]
If default error detection is enabled: the function shall check that the service xxx_Init was previously called. If the check fails, the function shall raise the default error XXX_E_NOT_INITIALIZED otherwise (if DET is disabled) return E_NOT_OK. ()"
- "In case default errors are enabled,..."
- "module raises the Default error XXX_E_TRANSITION"
- "The DET provides services to store default errors"
...

The correct text would be:
- sub chapter is called "7.x Development errors"
- "[SWS_xxx_yyyyy]
In case development error detection is enabled for the xxxx module: The xxxx module shall check API parameters for validity and report detected development errors to the DET. ()"
- "[SWS_xxx_yyyyy]
If development error detection is enabled: the function shall check that the service xxx_Init was previously called. If the check fails, the function shall raise the development error XXX_E_NOT_INITIALIZED otherwise (if DET is disabled) return E_NOT_OK. ()"
- "In case development errors are enabled,..."
- "module raises the development error XXX_E_TRANSITION"
- "The DET provides services to store development errors"

Solution for SWS_RTE:

– SWS_RTE —
- Change 4.8 Default errors to 4.8 Development errors
- Change "Errors which can occur at runtime in the RTE are classified as default errors" to "Errors which can occur at runtime in the RTE are classified as development errors"
- Remove [SWS_Rte_07676]
- Change [SWS_RTE_06611]"If a violation is detected the RTE shall report a default error to the DET." to "If a violation is detected the RTE shall report a development error to the DET."
- Change [SWS_Rte_06631]
[SWS_Rte_06631] d The RTE shall use the OS Application Identifier as the Instance Id to enable the developer to identify in which runtime section of the RTE the error occurs. This Instance ID is even unique across multi cores and so implicitly allows the development error to be traced to a specific core. c(SRS_BSW_00337)

SRS_Libraries:
- In chapter "3 Acronyms and abbreviations": Rename "Development Error Tracer" to "Default Error Tracer"

SRS_SPALGeneral:
- In chapter "6.1.1.3.1 [SRS_SPAL_00157] ...": Rename "Development Error Tracer" to "Default Error Tracer"
- In chapter "6.1.1.4.2 [SRS_SPAL_12448] ...": Rename "Development Error Tracer" to "Default Error Tracer"

SRS_FlashTest:
- In chapter "6.1 Functional Requirements": Rename "Development Error Tracer" to "Default Error Tracer"
- In chapter "7 References":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_MFXLibrary:
- In chapter "2 Acronyms and abbreviations": Rename "Development Error Tracer" to "Default Error Tracer"

SWS_MemoryAbstractionInterface:

- In chapter "3.1 Input documents":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_FlexRayNetworkManagement:
- In chapter "3.3 Related AUTOSAR documents":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_CANStateManager:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_PDURouter:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_EEPROMDriver:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"
–Last change on issue 73570 comment 47–

BW-C-Level:

Application	Specification	Bus
1	1	1