| Document Title | Specification for CAN XL Driver |
|---|---|
| **Document Category** | Software specification (SWS) |
| **Document UID** | 1014 |
| **Document Status** | published |
| **Part of AUTOSAR Standard** | Classic Platform |
| **Part of Standard Release** | R26-11 |

| Document Change History | | | |
|---|---|---|---|
| **Date** | **Release** | **Changed by** | **Description** |
| 2026-11-26 | R26-11 | AUTOSAR Release Management | • CANsec Support<br>• Support of zero-copy operation<br>• Editorial changes |
| 2025-11-27 | R25-11 | AUTOSAR Release Management | • Abstraction from Driver APIs<br>• Editorial changes |
| 2024-11-27 | R24-11 | AUTOSAR Release Management | • Introduce `CanXL_ReleaseRxBuffer()` and `CanXL_ImmediateTransmit()` as draft<br>• `CanXL_SetControllerMode()` change to Synchronous<br>• Fix Service IDs<br>• Update mandatory interfaces and imported types |
| 2023-11-23 | R23-11 | AUTOSAR Release Management | • Introduce `CanXL_GetCurrentTimeTuple()` as draft and deprecate `CanXL_GetCurrentTime()`<br>• Editorial changes<br>• Update optional interfaces<br>• Update imported datatypes |
| 2022-11-24 | R22-11 | AUTOSAR Release Management | • Initial release |

**Disclaimer**

This work (specification and/or software implementation) and the material contained in it, as released by AUTOSAR, is for the purpose of information only. AUTOSAR and the companies that have contributed to it shall not be liable for any use of the work.

The material contained in this work is protected by copyright and other types of intellectual property rights. The commercial exploitation of the material contained in this work requires a license to such intellectual property rights.

This work may be utilized or reproduced without any modification, in any form or by any means, for informational purposes only. For any other purpose, no part of the work may be utilized or reproduced, in any form or by any means, without permission in writing from the publisher.

The work has been developed for automotive applications only. It has neither been developed, nor tested for non-automotive applications.

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

# Table of Contents

1 Introduction and functional overview 6
2 Acronyms and Abbreviations 7
3 Related documentation 8
3.1 Input documents & related standards and norms . . . 8
3.2 Related specification . . . 8
4 Constraints and assumptions 9
4.1 Limitations . . . 9
5 Dependencies to other modules 10
5.1 File Structure . . . 10
5.1.1 Code File Structure . . . 10
6 Requirements Tracing 11
7 Functional specification 12
7.1 Initialization . . . 12
7.2 State Handling . . . 12
7.2.1 Communication Request . . . 12
7.2.2 BusOff Handling . . . 12
7.2.3 BusOff Handling without error signaling . . . 13
7.2.4 Wake Up . . . 14
7.3 Reception Handling . . . 14
7.4 Transmission Handling . . . 15
7.5 Error Classification . . . 16
7.5.1 Development Errors . . . 17
7.5.2 Runtime Errors . . . 17
7.5.3 Production Errors . . . 17
8 API specification 18
8.1 Imported types . . . 18
8.2 Type definitions . . . 19
8.2.1 CanXL_Params . . . 19
8.2.2 CanXL_PduType . . . 19
8.2.3 CanXL_HwType . . . 20
8.3 Function definitions . . . 20
8.3.1 CanXL_EnableEgressTimeStamp . . . 21
8.3.2 CanXL_GetControllerMode . . . 21
8.3.3 CanXL_GetCounterValues . . . 22
8.3.4 CanXL_GetCurrentTimeTuple . . . 23
8.3.5 CanXL_GetEgressTimeStamp . . . 25
8.3.6 CanXL_GetIngressTimeStamp . . . 26
8.3.7 CanXL_GetPhysAddr . . . 27

8.3.8 CanXL_GetRxStats . . . 27
8.3.9 CanXL_GetTxErrorCounterValues . . . 28
8.3.10 CanXL_GetTxStats . . . 29
8.3.11 CanXL_ProvideTxBuffer . . . 30
8.3.12 CanXL_ReleaseRxBuffer . . . 32
8.3.13 CanXL_SetControllerMode . . . 32
8.3.14 CanXL_SetPhysAddr . . . 33
8.3.15 CanXL_Transmit . . . 34
8.3.16 CanXL_ImmediateTransmit . . . 36
8.3.17 CanXL_UpdatePhysAddrFilter . . . 37
8.3.18 CanXL_Write . . . 38
8.3.19 CanXL_XLReleaseRxBuffer . . . 40
8.4 Scheduled functions . . . 41
8.5 Expected interfaces . . . 41
8.5.1 Mandatory interfaces . . . 41
8.5.2 Optional interfaces . . . 41
8.5.3 Callout interfaces . . . 42
8.5.4 Configurable interfaces . . . 42
9 Sequence diagrams 43
10 Configuration specification 44
10.1 How to read this chapter . . . 44
10.2 Containers and configuration parameters . . . 44
10.2.1 CanXLGeneral . . . 44
10.2.2 CanXLController . . . 46
10.2.3 CanXLCanSecConfig . . . 49
10.2.3.1 CanXLCanSecBypasses . . . 50
10.2.3.2 CanXLCanSecSecurityEntity . . . 50
10.2.3.3 CanXLCanSecSecureChannel . . . 51
10.2.3.4 CanXLCanSecFilterRange . . . 53
10.2.4 CanXLEthEgressFifo . . . 58
10.2.5 CanXLEthIngressFifo . . . 60
10.2.6 CanXLBaudrateConfig . . . 61
10.2.7 CanXLHardwareObject . . . 65
10.2.8 CanXLHwFilter . . . 67
10.3 Configuration Hints . . . 67
10.4 Published Information . . . 67
A Not applicable requirements 68
B Change History 69
B.1 Change History of this document according to AUTOSAR Release R22-11 69
B.1.1 Added Specification Items in R22-11 . . . 69
B.1.2 Changed Specification Items in R22-11 . . . 72

B.1.3 Deleted Specification Items in R22-11 . . . 72
B.1.4 Added Constraints in R22-11 . . . 72
B.1.5 Changed Constraints in R22-11 . . . 72
B.1.6 Deleted Constraints in R22-11 . . . 73
B.2 Change History of this document according to AUTOSAR Release R23-11 74
B.2.1 Added Specification Items in R23-11 . . . 74
B.2.2 Changed Specification Items in R23-11 . . . 74
B.2.3 Deleted Specification Items in R23-11 . . . 74
B.2.4 Added Constraints in R23-11 . . . 74
B.2.5 Changed Constraints in R23-11 . . . 75
B.2.6 Deleted Constraints in R23-11 . . . 75
B.3 Change History of this document according to AUTOSAR Release R24-11 76
B.3.1 Added Specification Items in R24-11 . . . 76
B.3.2 Changed Specification Items in R24-11 . . . 76
B.3.3 Deleted Specification Items in R24-11 . . . 76
B.3.4 Added Constraints in R24-11 . . . 76
B.3.5 Changed Constraints in R24-11 . . . 77
B.3.6 Deleted Constraints in R24-11 . . . 77
B.4 Change History of this document according to AUTOSAR Release R25-11 78
B.4.1 Added Specification Items in R25-11 . . . 78
B.4.2 Changed Specification Items in R25-11 . . . 78
B.4.3 Deleted Specification Items in R25-11 . . . 79
B.4.4 Added Constraints in R25-11 . . . 79
B.4.5 Changed Constraints in R25-11 . . . 79
B.4.6 Deleted Constraints in R25-11 . . . 80
B.5 Change History of this document according to AUTOSAR Release R26-11 81
B.5.1 Added Specification Items in R26-11 . . . 81
B.5.2 Changed Specification Items in R26-11 . . . 82
B.5.3 Deleted Specification Items in R26-11 . . . 84
B.5.4 Added Constraints in R26-11 . . . 84
B.5.5 Changed Constraints in R26-11 . . . 84
B.5.6 Deleted Constraints in R26-11 . . . 84

# 1 Introduction and functional overview

This specification describes the functionality, API and the configuration for the AUTOSAR Basic Software module `CAN XL Driver`.

The base for this document are [1, CiA610-1] and [2, CiA611-1]. It is assumed that the reader is familiar with these specifications. This document will not describe CAN XL functionality again.

The `CAN XL Driver` is part of the lowest layer, performs the hardware access and offers a hardware independent API to the upper layer. The two upper layers that have access to the `CAN XL Driver` are the `CanIf` and `EthIf` modules.

The `CAN XL Driver` is an extension of the `CAN Driver` module so this document shall only provide information and specifications which extends the existing CAN stack. Some general information is given for a better understanding.

The `CAN XL Driver` provides services for initiating transmissions and calls the call-back functions of the `CanIf` and `EthIf` modules for notifying events, independently from the hardware.

Furthermore, it provides services to control CAN XL Controller specific hardware including e.g. the transmission and reception of generic CAN XL frames.

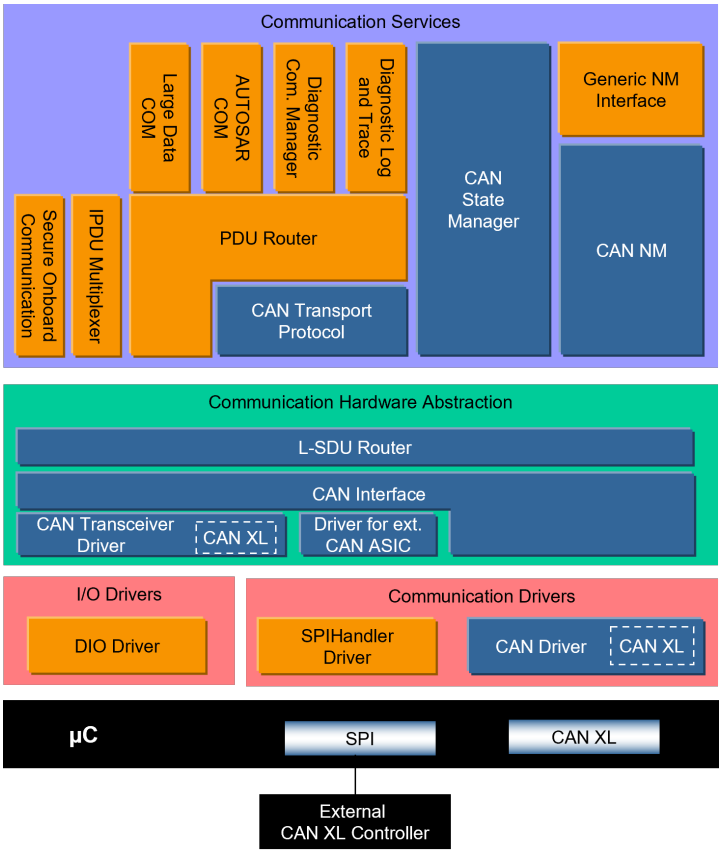


**Figure 1.1: AUTOSAR CAN XL Layered Architecture**

# 2 Acronyms and Abbreviations

The glossary below includes acronyms and abbreviations relevant to the `CAN XL Driver` module that are not included in the [3, AUTOSAR glossary].

| Abbreviation / Acronym: | Description: |
|---|---|
| XLFF | XL Frame Format |
| SDU type | service data unit type |
| retransmit counter | Feature of some CAN XL controllers that allows to send a CAN frame multiple times in a row in case the ACK slot is not set on the bus. The counter gives the maximum number of transmissions. |

# 3 Related documentation

## 3.1 Input documents & related standards and norms

[1] CiA 610-1 version 1.0.0 (DSP) - CAN XL specifications and test plans - Part 1: Data link layer and physical coding sub-layer requirements
http://www.can-cia.org

[2] CiA 611-1 version 1.0.0 (DSP) - CAN XL higher layer functions - Part 1: Definition of service data unit types
http://www.can-cia.org

[3] Glossary
AUTOSAR_FO_TR_Glossary

[4] General Specification of Basic Software Modules
AUTOSAR_CP_SWS_BSWGeneral

[5] Specification of CAN Driver
AUTOSAR_CP_SWS_CANDriver

[6] Specification of CAN XL Transceiver Driver
AUTOSAR_CP_SWS_CANXLTransceiverDriver

[7] Specification of CAN Interface
AUTOSAR_CP_SWS_CANInterface

[8] Specification of Ethernet Interface
AUTOSAR_CP_SWS_EthernetInterface

[9] General Requirements on Basic Software Modules
AUTOSAR_CP_RS_BSWGeneral

[10] ISO 11898-1:2015 – Road vehicles – Controller area network (CAN)

[11] Specification of Ethernet Driver
AUTOSAR_CP_SWS_EthernetDriver

## 3.2 Related specification

AUTOSAR provides a General Specification on Basic Software modules [4, SWS BSW General], which is also valid for `CAN XL Driver`.

Thus, the specification SWS BSW General shall be considered as additional and required specification for `CAN XL Driver`.

# 4 Constraints and assumptions

The constraints and assumptions of the CAN XL Driver are the same as for the CAN Driver module.

It is assumed, that all CAN XL Hardware supports out of the box both ingress and egress timestamping. Should this not be available, then it needs to be emulated in software.

## 4.1 Limitations

In [2, CiA611-1] there are several SDU Types specified which shall not directly supported by the AUTOSAR CAN XL stack. The solely directly in AUTOSAR communication stack supported SDU Types are as following:

1. 01h (content based CAN XL frames)
2. 03h (tunneled CAN 2.0/FD frames)
3. 05h (mapped tunneled 802.3 Ethernet frames)

Any other types like 02h (node addressing) and 04h (unmapped tunneled 802.3 Ethernet frames) shall not be directly supported. They still can be used with CDD, for details refer to CanXL_Write(), CanIf_TxConfirmation(), and CanIf_XLRxIndication(), and CanXL_XLReleaseRxBuffer() APIs.

Furthermore, future extensions making use of SEC bit of CAN XL Frame header like Security and Multi-PDU are currently in development and therefore not supported yet.

# 5 Dependencies to other modules

The `CAN XL Driver` module extends the `CAN Driver` [5] and has interfaces towards the [6, CAN XL Transceiver Driver], [5, CAN Driver], the [7, CAN Interface] and the [8, Ethernet Interface].

## 5.1 File Structure

This section explains the file structure of the `CAN XL Driver` module.

### 5.1.1 Code File Structure

For details, refer to the section 5.1.6 “Code file structure” in [4, SWS BSW General].

# 6 Requirements Tracing

The following tables reference the requirements specified in [9] and link to the fulfillment of these. Please note that if column “Satisfied by” is empty for a specific requirement this means that this requirement is not fulfilled by this document.

| Requirement | Description | Satisfied by |
|---|---|---|
| **[SRS_BSW_00312]** | Shared code shall be reentrant | [CP_SWS_CanXL_00104] |
| **[SRS_BSW_00323]** | All AUTOSAR Basic Software Modules shall check passed API parameters for validity | [CP_SWS_CanXL_00107] [CP_SWS_CanXL_00108] [CP_SWS_CanXL_00109] [CP_SWS_CanXL_00110] [CP_SWS_CanXL_00112] [CP_SWS_CanXL_00116] [CP_SWS_CanXL_00119] |
| **[SRS_BSW_00369]** | All AUTOSAR Basic Software Modules shall not return specific development error codes via the API | [CP_SWS_CanXL_00106] [CP_SWS_CanXL_00107] [CP_SWS_CanXL_00108] [CP_SWS_CanXL_00109] [CP_SWS_CanXL_00110] [CP_SWS_CanXL_00112] [CP_SWS_CanXL_00116] [CP_SWS_CanXL_00119] [CP_SWS_CanXL_00129] |
| **[SRS_Eth_00172]** | Ethernet Driver hardware supported data transfer | [SWS_CanXL_91000] |
| **[SRS_Eth_00173]** | Ethernet Driver transmission requests with direct data provision | [SWS_CanXL_91002] |
| **[SRS_Eth_00188]** | Ethernet Driver transmission requests with indirect data provision | [CP_SWS_CanXL_00139] |

**Table 6.1: Requirements Tracing**

# 7 Functional specification

## 7.1 Initialization

**[CP_SWS_CanXL_00001]** ⌈The `Can_Init()` shall be extended by all functionality necessary to properly initialize the `CAN XL Driver`.⌋

## 7.2 State Handling

### 7.2.1 Communication Request

State handling is performed individually per bus, for native CAN by `CanSM` and for tunneled Ethernet via CAN XL by `EthSM`. The `EthIf` does directly communicate with `CAN XL Driver` and `CAN XL Transceivers` as lower layer.

The actual physical bus state is only controlled by the CAN stack. The `CAN XL Transceiver` indicates the physical bus state through LinkState polling to the Ethernet stack, independently of the Ethernet controller mode.

The Ethernet controller mode is stored by the `CAN XL Driver`, and returned on request.

**[CP_SWS_CanXL_00113]** ⌈The Ethernet controller mode shall initially be set to `ETH_MODE_DOWN`.⌋

**[CP_SWS_CanXL_00002]** ⌈The controller mode requested by the Ethernet stack via `CanXL_SetControllerMode()` shall have no effect on the CAN XL controller hardware. The new mode shall be solely stored.⌋

**[CP_SWS_CanXL_00114]** ⌈`CanXL_GetControllerMode()` shall return the stored mode.⌋

**[CP_SWS_CanXL_00128]** ⌈When the Ethernet controller mode changes, the `CAN XL Driver` shall report this change via `EthIf_CtrlModeIndication()`.⌋

### 7.2.2 BusOff Handling

There always needs to be a CAN Stack configured even in case no native CAN communication is used and `BusOff Handling` is performed as defined in `CAN Driver`.

**[CP_SWS_CanXL_00003]** ⌈Changes of the bus error state shall be notified to `CAN XL Transceiver` by a call to `CanIf_XLReportErrorState()`.⌋

The parameter `Transceiver` of `CanIf_XLReportErrorState()` can be obtained by linking the `CanXLController` via `CanIfCtrlCfg` and `CanSMManagerNetwork`

to CanIfTrcvCfg. All buffers of the affected CAN XL controller are flushed within CAN XL Driver if a BusOff occurs.

**[CP_SWS_CanXL_00004]** ⌈In case an Ethernet stack is configured, also the buffers containing frames for Ethernet shall be flushed and EthIf_TxConfirmation() shall be called with result E_NOT_OK at the event of a BusOff.⌋

EthIf is polling its hardware if the current mode and link state match the requested mode and link state. A BusOff event is returned by CAN XL Transceiver to EthIf by reporting link down in context of CanXLTrcv_GetLinkState(). Furthermore, the link status which is present in the EthIf does not differentiate between error active and error passive state, they both are reported as link up. See [6, CAN XL Transceiver Driver] for further information.

### 7.2.3 BusOff Handling without error signaling

In case transceiver mode switching is used, error signaling must be turned off for currently existing transceivers. Without error signaling, there also won't be any busoff handling by the controller hardware itself. Therefore, in this case, there shall be a simple implementation in CAN XL Driver to still perform babbling protection.

**[CP_SWS_CanXL_00005]** ⌈If error signaling is disabled, a basic CAN busoff handling with TEC (Transmission Error Counter) and REC (Reception Error Counter) shall be emulated in software.⌋

The following example shows how this handling would typically look like:

TEC is normally a counter initialized with 0 saturating in range of 0 to 256, and REC is a counter initialized with 0 saturating in range of 0 to 128. The retransmit counter is turned off completely or at least configured to a very low value.

There would be a state machine following this basic rule set:

- When a frame is transmitted but no ACK slot was set on the bus, TEC shall be increased by 8. TEC shall not be increased beyond 128 before at least one frame has been received on the bus since the last startup or bus-off recovery. It does not matter whether this frame was received successfully or not.
- When a frame is transmitted and the ACK slot was set on the bus, TEC shall be decreased by 1.
- When a frame is received but is not consistent, REC shall be increased by 1.
- When a frame is received consistent, REC shall be decreased by 1.

The detection of these events would be performed alongside the transmission and reception handling in their respective context (MainFunction or ISR) and additionally, depending on the hardware capabilities, in bus-off context for the error events.

The state transitions are expected to correspond to those defined in [10, ISO 11898-1:2015] chapter 12.1.4.4 "Bus-off management". In bus-off state, TEC and REC are reset immediately. Basically, the CAN XL Driver state transitions should not differ when error signaling is disabled from the handling when error signaling is available.

### 7.2.4 Wake Up

Basic wakeup handling does not change.

**[CP_SWS_CanXL_00006]** ⌈There shall be no differentiation with regards to SDU-Type of the received frame triggering wakeup; any frame received by CAN XL Driver shall notify a CAN wakeup.⌋

EcuM_SetWakeupEvent() is solely called for Can. It is never notified for Ethernet, the CAN XL Transceiver indicates the physical bus state through LinkState polling to the Ethernet stack instead.

## 7.3 Reception Handling

**[CP_SWS_CanXL_00007]** ⌈All used reception queues for CAN XL frames shall be mapped individually to CAN XL HRHs.⌋

Reception queues will also be used for CAN 2.0 and CAN FD frames mapped to CAN HRHs.

The reception mechanism of SDU Type shall be extended for XLFF, e.g. in polling mode Can_MainFunction_Read() shall be extended. See chapter "L-PDU reception" of CAN Driver for details.

**[CP_SWS_CanXL_00008]** ⌈On L-PDU reception of XLFF, first any configured filtering shall be performed before other functionality is performed.⌋

**[CP_SWS_CanXL_00009]** ⌈On L-PDU reception of XLFF, if SDU Type equals 05h (mapped tunneled 802.3 Ethernet frames) the CAN XL Driver shall extract payload and addressing information from the received frame and pass them to EthIf_Rx-Indication(); otherwise it shall pass the received frame directly to CanIf_XLRx-Indication().⌋

In either case, the CAN XL driver keeps the reception buffer until CanXL_ReleaseRx-Buffer() is called to enable asynchronous handling of the buffer in the upper layers.

**[CP_SWS_CanXL_00140]** ⌈The receive buffer shall be freed upon a call to CanXL_ReleaseRxBuffer() (SDU Type equals 05h) or CanXL_XLReleaseRxBuffer() (other SDU Types).⌋

Please note that the call to CanXL_ReleaseRxBuffer() and CanXL_XLRelease-RxBuffer() may also occur in the context of the EthIf_RxIndication() and CanIf_XLRxIndication(), respectively.

Depending on available filtering on hardware support, there might be multiple reception queues available to filter received frames into.

As this is highly hardware vendor specific, there shall be no details given here beyond following:

Any field available in the CAN XL Frame Header might be used for filtering, it is not restricted to the Acceptance Field only. Purpose might be separation of traffic classes e.g. based on Priority ID, VCID and/or SDU Type.

When it comes to tunneling of ethernet frames, a common use case would be to filter any received SDU Type 05h (mapped tunneled 802.3 Ethernet frames) L-PDU for matching ethernet MAC address configured (see CanXLEthPhysAddress).

It can help to handle first high priority traffic before lower priority traffic. Also it would be possible to put unwanted traffic in a separate queue to keep for further inspection instead of filtering it out.

**[CP_SWS_CanXL_00117]** ⌈On L-PDU reception of XLFF, the frame shall be checked for consistency with CiA 611-1 chapter 5 SDU types specification.⌋

**[CP_SWS_CanXL_00118]** ⌈If the consistency check fails the received frame shall be discarded and a runtime error CANXL_E_INV_DATA shall be reported.⌋

## 7.4 Transmission Handling

As CAN XL is designed to scale from small to big systems and the payload range goes up to 2048 bytes, the specific hardware implementations may differ.

The use of CAN XL HTH is similar to CAN HTHs. See chapter "L-PDU transmission" of CAN Driver for details.

The CAN HTH is used for mapping to logical transmission objects. Depending on available hardware features these might map to separate message objects or hardware supported queues.

**[CP_SWS_CanXL_00010]** ⌈All used transmission queues for CAN XL frames shall be mapped individually to CAN XL HTHs.⌋

CAN XL transmission queues could also have a size of one and therefore correspond to traditional CAN hardware objects. Transmission queues for CAN 2.0 and CAN FD frames will be mapped similarly to CAN HTHs.

As the availability and behavior of hardware supported queues is is highly hardware vendor specific, there shall be no details given here. It is the responsibility of the

CAN XL Driver vendor to document how the different CAN XL HTH behave. For same hardware there might be several mappings implemented depending of the specific needs of the system.

**[CP_SWS_CanXL_00122]** ⌈The functions CanXL_Transmit() (following CanXL_ProvideTxBuffer()) and CanXL_ImmediateTransmit() (for SDU Type 05h) and CanXL_Write() (for other SDU Types) shall transfer the XLFF control information and data to hardware and shall request the start of transmission.⌋

**[CP_SWS_CanXL_00141]** ⌈If asynchronous buffer semantics is enabled (parameter AsyncBuffers of CanXL_ImmediateTransmit() is set to TRUE for SDU Type 05h, or parameter AsyncBuffer of CanXL_Write() is set to TRUE for other SDU Types), the transmit buffer shall be used directly in the hardware descriptor. It is freed by the call to EthIf_TxConfirmation() or CanIf_TxConfirmation(), respectively.⌋

**[CP_SWS_CanXL_00011]** ⌈After successful transmission of XLFF on the bus, if SDU Type of the original transmission equals 05h (mapped tunneled 802.3 Ethernet frames) the CAN XL Driver shall call EthIf_TxConfirmation() with the result E_OK, else CanIf_TxConfirmation() shall be called.⌋

**[CP_SWS_CanXL_00012]** ⌈In case a transmission can't be performed successfully, if SDU Type of the original transmission equals 05h (mapped tunneled 802.3 Ethernet frames) the CAN XL Driver shall call EthIf_TxConfirmation() with the result E_NOT_OK.⌋

**[CP_SWS_CanXL_00013]** ⌈In case a CAN XL controller is in another state than CAN_CS_STARTED, all buffers of the affected CAN XL controller shall be flushed.⌋

## 7.5 Error Classification

Chapter [4, General Specification of Basic Software Modules] 7.2 *"Error Handling"* describes the error handling of the Basic Software in detail. Above all, it constitutes a classification scheme consisting of five error types which may occur in BSW modules. Based on this foundation, the following section specifies particular errors arranged in the respective subsections below.

### 7.5.1 Development Errors

**[CP_SWS_CanXL_10024] Definition of development errors in module CanXL** ⌈

| *Type of error* | *Related error code* | *Error value* |
|---|---|---|
| API Service called with wrong parameter | CANXL_E_PARAM_POINTER | 0x01 |
| API Service called with wrong parameter | CANXL_E_PARAM_HANDLE | 0x02 |
| API Service called with wrong parameter | CANXL_E_PARAM_DATA_LENGTH | 0x03 |
| API Service called with wrong parameter | CANXL_E_PARAM_CONTROLLER | 0x04 |
| API Service used without initialization | CANXL_E_UNINIT | 0x05 |
| Invalid parameter | CANXL_E_INV_PARAM | 0x10 |
| Invalid mode | CANXL_E_INV_MODE | 0x11 |
| Invalid clock unit index | CANXL_E_INV_CLKUNIT_IDX | 0x12 |

⌋

### 7.5.2 Runtime Errors

**[CP_SWS_CanXL_10025] Definition of runtime errors in module CanXL** ⌈

| *Type of error* | *Related error code* | *Error value* |
|---|---|---|
| Invalid data | CANXL_E_INV_DATA | 0x12 |

⌋

### 7.5.3 Production Errors

There are no addtional production errors.

# 8 API specification

Please note, that the `CAN XL Driver` uses the `MSN Can` for parts that are shared with the [5, CAN Driver] and the `MSN CanXL` for extensions defined in this document. Deviating from SRS_BSW_00101 and SRS_BSW_00407, the `CAN XL Driver` does not provide separate `Init` and `GetVersionInfo` APIs with the `MSN CanXL`. Following SWS_MemMap_00022, memory sections associated with APIs defined in this document will use the `MSN CanXL`, and also symbolic name values referring to containers defined in this document will use the `MSN CanXL` to follow TPS_ECUC_02108.

## 8.1 Imported types

In this chapter all types included from the following files are listed.

**[CP_SWS_CanXL_10023] Definition of imported datatypes of module CanXL** ⌈

| *Module* | *Header File* | *Imported Type* |
|---|---|---|
| Can | Can_GeneralTypes.h | Can_ErrorStateType |
| | Can_GeneralTypes.h | Can_HwHandleType |
| Comtype | ComStack_Types.h | BufReq_ReturnType |
| | ComStack_Types.h | ListElemStructType (draft) |
| | ComStack_Types.h | PduIdType |
| | ComStack_Types.h | PduInfoType |
| | ComStack_Types.h | PduLengthType |
| | ComStack_Types.h | TimeStampQualType (draft) |
| | ComStack_Types.h | TimeStampType (draft) |
| | ComStack_Types.h | TimeTupleType (draft) |
| Eth | Eth_GeneralTypes.h | Eth_BufIdxType |
| | Eth_GeneralTypes.h | Eth_CounterType |
| | Eth_GeneralTypes.h | Eth_FilterActionType |
| | Eth_GeneralTypes.h | Eth_FrameType |
| | Eth_GeneralTypes.h | Eth_ModeType |
| | Eth_GeneralTypes.h | Eth_RxStatsType |
| | Eth_GeneralTypes.h | Eth_TxErrorCounterValuesType |
| | Eth_GeneralTypes.h | Eth_TxStatsType |
| Std | Std_Types.h | Std_ReturnType |

⌋

## 8.2 Type definitions

### 8.2.1 CanXL_Params

**[CP_SWS_CanXL_10001] Definition of datatype CanXL_Params** ⌈

| | | |
|---|---|---|
| ***Name*** | CanXL_Params | |
| ***Kind*** | Structure | |
| ***Elements*** | PriorityId | |
| | ***Type*** | uint16 |
| | ***Comment*** | Priority ID of a CAN XL message. |
| | Vcid | |
| | ***Type*** | uint16 |
| | ***Comment*** | VCID of a CAN XL message. |
| | SduType | |
| | ***Type*** | uint8 |
| | ***Comment*** | SDU type of a CAN XL message. |
| | AcceptanceField | |
| | ***Type*** | uint32 |
| | ***Comment*** | Acceptance field of a CAN XL message. |
| | Sec | |
| | ***Type*** | uint8 |
| | ***Comment*** | Simple extended content field of a CAN XL message. |
| ***Description*** | Contains CAN XL specific information. | |
| ***Available via*** | Can_GeneralTypes.h | |

⌋

### 8.2.2 CanXL_PduType

**[CP_SWS_CanXL_10026] Definition of datatype CanXL_PduType** ⌈

| | | |
|---|---|---|
| ***Name*** | CanXL_PduType | |
| ***Kind*** | Structure | |
| ***Elements*** | swPduHandle | |
| | ***Type*** | PduIdType |
| | ***Comment*** | Contains the PDU ID. |
| | length | |
| | ***Type*** | uint16 |
| | ***Comment*** | Length of the data. |
| | sdu | |
| | ***Type*** | uint8* |
| | ***Comment*** | SDU data pointer. |
| | XLParams | |
| | ***Type*** | CanXL_Params* |

▽

△

| | | |
|---|---|---|
| | ***Comment*** | Pointer to CAN XL params. |
| ***Description*** | This type extends the classical Can_PduType with a larger PDU length, the CanXL_Params and a sec to indicate simple or extended content. | |
| ***Available via*** | Can_GeneralTypes.h | |

⌋

### 8.2.3 CanXL_HwType

**[CP_SWS_CanXL_10027] Definition of datatype CanXL_HwType** ⌈

| | | |
|---|---|---|
| ***Name*** | CanXL_HwType | |
| ***Kind*** | Structure | |
| ***Elements*** | XLParams | |
| | ***Type*** | CanXL_Params* |
| | ***Comment*** | Pointer to CAN XL params. |
| | ControllerId | |
| | ***Type*** | uint8 |
| | ***Comment*** | ControllerId provided by CanIf, identifies the corresponding CAN XL controller. |
| | Hoh | |
| | ***Type*** | Can_HwHandleType |
| | ***Comment*** | ID of the corresponding CAN XL Hardware Object Range. |
| ***Description*** | This type defines a data structure which provides a CAN XL Hardware Object Handle including its corresponding CAN Controller and therefore CanDrv as well as the specific CAN XL parameters. | |
| ***Available via*** | Can_GeneralTypes.h | |

⌋

## 8.3 Function definitions

This chapter contains all APIs provided by the module.

Note: `CAN XL Driver` does not have additional callback notifications.

### 8.3.1 CanXL_EnableEgressTimeStamp

**[CP_SWS_CanXL_10004] Definition of API function CanXL_EnableEgressTime Stamp** ⌈

| | | |
|---|---|---|
| ***Service Name*** | CanXL_EnableEgressTimeStamp | |
| ***Syntax*** | `void CanXL_EnableEgressTimeStamp ( uint8 CtrlIdx, Eth_BufIdxType BufIdx )` | |
| ***Service ID [hex]*** | 0x17 | |
| ***Sync/Async*** | Synchronous | |
| ***Reentrancy*** | Non Reentrant | |
| ***Parameters (in)*** | CtrlIdx | Index of the addresses controller. |
| | BufIdx | Index of the message buffer, where Application expects egress time stamping |
| ***Parameters (inout)*** | None | |
| ***Parameters (out)*** | None | |
| ***Return value*** | None | |
| ***Description*** | Activates egress time stamping on a dedicated message object. Some HW does store once the egress time stamp marker and some HW needs it always before transmission. There will be no "disable" functionality, due to the fact, that the message type is always "time stamped" by network design. | |
| ***Available via*** | CanXL.h | |

⌋

Note: This API is derived from Ethernet Driver ([SWS_Eth_00186]). For better understanding of the API's original intention you may check [8, Ethernet Interface] and [11, Ethernet Driver]. It is called by the AUTOSAR Ethernet Interface to achieve compatible CAN XL agnostic behavior.

**[CP_SWS_CanXL_00123]** ⌈The service `CanXL_EnableEgressTimeStamp()` has no functionality and shall return without performing any action.⌋

### 8.3.2 CanXL_GetControllerMode

**[CP_SWS_CanXL_10017] Definition of API function CanXL_GetControllerMode** ⌈

| | | |
|---|---|---|
| ***Service Name*** | CanXL_GetControllerMode | |
| ***Syntax*** | `Std_ReturnType CanXL_GetControllerMode ( uint8 CtrlIdx, Eth_ModeType* CtrlModePtr )` | |
| ***Service ID [hex]*** | 0x1b | |
| ***Sync/Async*** | Synchronous | |
| ***Reentrancy*** | Non Reentrant | |
| ***Parameters (in)*** | CtrlIdx | Index of the controller within the context of the Driver |
| ***Parameters (inout)*** | None | |

▽

△

| | | |
|---|---|---|
| ***Parameters (out)*** | CtrlModePtr | ETH_MODE_DOWN: the Rx/Tx communication of the controller is disabled<br>ETH_MODE_ACTIVE: the Rx/Tx communication of the controller is enabled |
| ***Return value*** | Std_ReturnType | `E_OK`: success<br>`E_NOT_OK`: controller mode could not be obtained |
| ***Description*** | Obtains the communication state of the indexed controller | |
| ***Available via*** | CanXL.h | |

⌋

Note: This API is derived from Ethernet Driver ([SWS_Eth_91010]). For better understanding of the API's original intention you may check [8, Ethernet Interface] and [11, Ethernet Driver].

**[CP_SWS_CanXL_00046]** ⌈The function shall read the current communication state of the indexed controller.⌋

The current communication state is set as described in [CP_SWS_CanXL_00113] and [CP_SWS_CanXL_00002].

**[CP_SWS_CanXL_00047]** ⌈If development error detection is enabled: `CanXL_GetControllerMode()` shall check the parameter `CtrlIdx` for being valid. If the check fails, the function shall raise the development error `CANXL_E_PARAM_CONTROLLER`.⌋

**[CP_SWS_CanXL_00129]**

*Upstream requirements:* SRS_BSW_00369

⌈If development error detection is enabled: `CanXL_GetControllerMode()` shall raise the error `CANXL_E_UNINIT` if the driver is not yet initialized.⌋

### 8.3.3 CanXL_GetCounterValues

**[CP_SWS_CanXL_10005] Definition of API function CanXL_GetCounterValues** ⌈

| | | |
|---|---|---|
| ***Service Name*** | CanXL_GetCounterValues | |
| ***Syntax*** | `Std_ReturnType CanXL_GetCounterValues ( uint8 CtrlIdx, Eth_CounterType* CounterPtr )` | |
| ***Service ID [hex]*** | 0x14 | |
| ***Sync/Async*** | Synchronous | |
| ***Reentrancy*** | Non Reentrant | |
| ***Parameters (in)*** | CtrlIdx | Index of the controller within the context of the Driver |
| ***Parameters (inout)*** | None | |
| ***Parameters (out)*** | CounterPtr | counter values according to IETF RFC 1757, RFC 1643 and RFC 2233. |

▽

△

| | | |
|---|---|---|
| ***Return value*** | Std_ReturnType | `E_OK`: success<br>`E_NOT_OK`: counter values read failure |
| ***Description*** | Reads a list with drop counter values of the corresponding controller. The meaning of these values is described at Eth_CounterType. | |
| ***Available via*** | CanXL.h | |

⌋

Note: This API is derived from Ethernet Driver ([SWS_Eth_00226]). For better understanding of the API's original intention you may check [8, Ethernet Interface] and [11, Ethernet Driver]. It is called by the AUTOSAR Ethernet Interface to achieve compatible CAN XL agnostic behavior.

**[CP_SWS_CanXL_00048]** ⌈The service `CanXL_GetCounterValues()` has no functionality and shall always return E_NOT_OK without performing any action.⌋

### 8.3.4 CanXL_GetCurrentTimeTuple

**[CP_SWS_CanXL_90000] Definition of API function CanXL_GetCurrentTimeTuple**

*Status:* DRAFT

⌈

| | | |
|---|---|---|
| ***Service Name*** | CanXL_GetCurrentTimeTuple (draft) | |
| ***Syntax*** | `Std_ReturnType CanXL_GetCurrentTimeTuple (`<br>`  uint8 CtrlIdx,`<br>`  uint8 ClkUnitIdx,`<br>`  TimeTupleType currentTimeTuplePtr`<br>`)` | |
| ***Service ID [hex]*** | 0x22 | |
| ***Sync/Async*** | Synchronous | |
| ***Reentrancy*** | Non Reentrant | |
| ***Parameters (in)*** | CtrlIdx | Index of the addressed controller |
| | ClkUnitIdx | Index of the addressed clock unit to provide the time tuple |
| ***Parameters (inout)*** | None | |
| ***Parameters (out)*** | currentTimeTuplePtr | Current time provided as time tuple |
| ***Return value*** | Std_ReturnType | `E_OK`: Current time successfully provided<br>`E_NOT_OK`:Current time not available |
| ***Description*** | Reads the time tuple of the current time of the timestamp clock and the current time of the PHC in an atomic operation. If no PHC is supported, the PHC value will be a copy of the timestamp clock value.<br>**Tags:** atp.Status=draft | |
| ***Available via*** | CanXL.h | |

⌋

Note: This API is derived from Ethernet Driver ([SWS_Eth_91017]). For better understanding of the API's original intention you may check [8, Ethernet Interface] and [11, Ethernet Driver].

**[CP_SWS_CanXL_00133]**

*Status:* DRAFT

⌈If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error CANXL_E_PARAM_CONTROLLER.⌋

**[CP_SWS_CanXL_00134]**

*Status:* DRAFT

⌈If development error detection is enabled: the function shall check the parameter ClkUnitIdx for being valid. If the check fails, the function shall raise the development error CANXL_E_INV_CLKUNIT_IDX.⌋

**[CP_SWS_CanXL_00135]**

*Status:* DRAFT

⌈If development error detection is enabled: the function shall check the parameter currentTimeTuplePtr and currentTimeTuplePtr for being valid. If the check fails, the function shall raise the development error CANXL_E_PARAM_POINTER.⌋

**[CP_SWS_CanXL_00136]**

*Status:* DRAFT

⌈The function shall be pre compile time configurable On/Off by the configuration parameter CanXLEthGlobalTimeSupport.⌋

**[CP_SWS_CanXL_00137]**

*Status:* DRAFT

⌈If development error detection is enabled: CanXL_GetCurrentTimeTuple() shall raise the error CANXL_E_UNINIT if the driver is not yet initialized.⌋

In case the Com-Stack is distributed across several partitions, the Can/Ethernet stack could reside in a different partition than the StbM module calling CanXL_GetCurrentTimeTuple() (via EthIf_GetCurrentTimeTuple()) API, means the call of CanXL_GetCurrentTimeTuple() could happen in another partition.

### 8.3.5 CanXL_GetEgressTimeStamp

**[CP_SWS_CanXL_10007] Definition of API function CanXL_GetEgressTimeStamp** ⌈

| | | |
|---|---|---|
| ***Service Name*** | CanXL_GetEgressTimeStamp | |
| ***Syntax*** | `Std_ReturnType CanXL_GetEgressTimeStamp ( uint8 CtrlIdx, Eth_BufIdxType BufIdx, TimeStampQualType* timeQualPtr, TimeStampType* timeStampPtr )` | |
| ***Service ID [hex]*** | 0x18 | |
| ***Sync/Async*** | Synchronous | |
| ***Reentrancy*** | Non Reentrant | |
| ***Parameters (in)*** | CtrlIdx | Index of the addresses controller. |
| | BufIdx | Index of the message buffer, where Application expects egress time stamping |
| ***Parameters (inout)*** | None | |
| ***Parameters (out)*** | timeQualPtr | quality of HW time stamp, e.g. based on current drift |
| | timeStampPtr | current time stamp |
| ***Return value*** | Std_ReturnType | `E_OK`: success<br>`E_NOT_OK`: failed to read time stamp. |
| ***Description*** | Reads back the egress time stamp on a dedicated message object. It must be called within the TxConfirmation() function. | |
| ***Available via*** | CanXL.h | |

⌋

Note: This API is derived from Ethernet Driver ([SWS_Eth_00190]). For better understanding of the API's original intention you may check [8, Ethernet Interface] and [11, Ethernet Driver].

**[CP_SWS_CanXL_00031]** ⌈If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error `CANXL_E_PARAM_CONTROLLER`.⌋

**[CP_SWS_CanXL_00032]** ⌈If development error detection is enabled: the function shall check the parameter timeQualPtr and timeStampPtr for being valid. If the check fails, the function shall raise the development error `CANXL_E_PARAM_POINTER`.⌋

**[CP_SWS_CanXL_00124]** ⌈The function shall be pre compile time configurable On/Off by the configuration parameter `CanXLEthGlobalTimeSupport`.⌋

**[CP_SWS_CanXL_00034]** ⌈If development error detection is enabled: `CanXL_GetEgressTimeStamp()` shall raise the error `CANXL_E_UNINIT` if the driver is not yet initialized.⌋

### 8.3.6 CanXL_GetIngressTimeStamp

**[CP_SWS_CanXL_10008] Definition of API function CanXL_GetIngressTimeStamp**

*Status:* OBSOLETE

⌈

| | | |
|---|---|---|
| ***Service Name*** | CanXL_GetIngressTimeStamp (obsolete) | |
| ***Syntax*** | `Std_ReturnType CanXL_GetIngressTimeStamp ( uint8 CtrlIdx, const uint8* DataPtr, TimeStampQualType* timeQualPtr, TimeStampType* timeStampPtr )` | |
| ***Service ID [hex]*** | 0x19 | |
| ***Sync/Async*** | Synchronous | |
| ***Reentrancy*** | Non Reentrant | |
| ***Parameters (in)*** | CtrlIdx | Index of the addresses controller. |
| | DataPtr | Pointer to the message buffer, where Application expects ingress time stamping |
| ***Parameters (inout)*** | None | |
| ***Parameters (out)*** | timeQualPtr | quality of HW time stamp, e.g. based on current drift |
| | timeStampPtr | current time stamp |
| ***Return value*** | Std_ReturnType | `E_OK`: success<br>`E_NOT_OK`: failed to read time stamp. |
| ***Description*** | Reads back the ingress time stamp on a dedicated message object. It must be called within the RxIndication() function.<br>**Tags:** atp.Status=obsolete | |
| ***Available via*** | CanXL.h | |

⌋

Note: This API is derived from Ethernet Driver. For better understanding of the API's original intention you may check [8, Ethernet Interface] and [11, Ethernet Driver].

**[CP_SWS_CanXL_00036]** ⌈If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error `CANXL_E_PARAM_CONTROLLER`.⌋

**[CP_SWS_CanXL_00037]** ⌈If development error detection is enabled: the function shall check the parameter DataPtr, timeQualPtr and timeStampPtr for being valid. If the check fails, the function shall raise the development error `CANXL_E_PARAM_POINTER`.⌋

**[CP_SWS_CanXL_00125]** ⌈The function shall be pre compile time configurable On/Off by the configuration parameter `CanXLEthGlobalTimeSupport`.⌋

**[CP_SWS_CanXL_00039]** ⌈If development error detection is enabled: `CanXL_GetIngressTimeStamp()` shall raise the error `CANXL_E_UNINIT` if the driver is not yet initialized.⌋

### 8.3.7 CanXL_GetPhysAddr

**[CP_SWS_CanXL_10018] Definition of API function CanXL_GetPhysAddr** ⌈

| | | |
|---|---|---|
| ***Service Name*** | CanXL_GetPhysAddr | |
| ***Syntax*** | `void CanXL_GetPhysAddr ( uint8 CtrlIdx, uint8* PhysAddrPtr )` | |
| ***Service ID [hex]*** | 0x25 | |
| ***Sync/Async*** | Synchronous | |
| ***Reentrancy*** | Non Reentrant | |
| ***Parameters (in)*** | CtrlIdx | Index of the controller within the context of the Driver |
| ***Parameters (inout)*** | None | |
| ***Parameters (out)*** | PhysAddrPtr | Physical source address (MAC address) in network byte order. |
| ***Return value*** | void | None |
| ***Description*** | Obtains the physical source address used by the indexed controller | |
| ***Available via*** | CanXL.h | |

⌋

Note: This API is derived from Ethernet Driver ([SWS_Eth_00052]). For better understanding of the API's original intention you may check [8, Ethernet Interface] and [11, Ethernet Driver].

**[CP_SWS_CanXL_00040]** ⌈The function shall read the source address used by the indexed controller (see `CanXLEthPhysAddress`).⌋

**[CP_SWS_CanXL_00042]** ⌈If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error `CANXL_E_PARAM_CONTROLLER`.⌋

**[CP_SWS_CanXL_00043]** ⌈If development error detection is enabled: the function shall check the parameter PhysAddrPtr for being valid. If the check fails, the function shall raise the development error `CANXL_E_PARAM_POINTER`.⌋

**[CP_SWS_CanXL_00044]** ⌈If development error detection is enabled: `CanXL_GetPhysAddr()` shall raise the error `CANXL_E_UNINIT` if the driver is not yet initialized.⌋

### 8.3.8 CanXL_GetRxStats

**[CP_SWS_CanXL_10009] Definition of API function CanXL_GetRxStats** ⌈

| | |
|---|---|
| ***Service Name*** | CanXL_GetRxStats |
| ***Syntax*** | `Std_ReturnType CanXL_GetRxStats ( uint8 CtrlIdx, Eth_RxStatsType* RxStats )` |

▽

△

| | | |
|---|---|---|
| ***Service ID [hex]*** | 0x15 | |
| ***Sync/Async*** | Synchronous | |
| ***Reentrancy*** | Non Reentrant | |
| ***Parameters (in)*** | CtrlIdx | Index of the controller within the context of the Driver |
| ***Parameters (inout)*** | None | |
| ***Parameters (out)*** | RxStats | List of values according to IETF RFC 2819 (Remote Network Monitoring Management Information Base) |
| ***Return value*** | Std_ReturnType | `E_OK`: success<br>`E_NOT_OK`: drop counter could not be obtained |
| ***Description*** | Returns the following list according to IETF RFC2819, where the maximal possible value shall denote an invalid value, e.g. if this counter is not available: 1. etherStatsDropEvents 2. ether StatsOctets 3. etherStatsPkts 4. etherStatsBroadcastPkts 5. etherStatsMulticastPkts 6. ether StatsCrcAlignErrors 7. etherStatsUndersizePkts 8. etherStatsOversizePkts 9. etherStats Fragments 10. etherStatsJabbers 11. etherStatsCollisions 12. etherStatsPkts64Octets 13. etherStatsPkts65to127Octets 14. etherStatsPkts128to255Octets 15. etherStatsPkts256to511 Octets 16. etherStatsPkts512to1023Octets 17. etherStatsPkts1024to1518Octets | |
| ***Available via*** | CanXL.h | |

⌋

Note: This API is derived from Ethernet Driver ([SWS_Eth_00233]). For better understanding of the API's original intention you may check [8, Ethernet Interface] and [11, Ethernet Driver]. It is called by the AUTOSAR Ethernet Interface to achieve compatible CAN XL agnostic behavior.

**[CP_SWS_CanXL_00020]** ⌈The service `CanXL_GetRxStats()` has no functionality and shall always return E_NOT_OK without performing any action.⌋

### 8.3.9 CanXL_GetTxErrorCounterValues

**[CP_SWS_CanXL_10010] Definition of API function CanXL_GetTxErrorCounter Values** ⌈

| | | |
|---|---|---|
| ***Service Name*** | CanXL_GetTxErrorCounterValues | |
| ***Syntax*** | `Std_ReturnType CanXL_GetTxErrorCounterValues ( uint8 CtrlIdx, Eth_TxErrorCounterValuesType* TxErrorCounterValues )` | |
| ***Service ID [hex]*** | 0x1d | |
| ***Sync/Async*** | Synchronous | |
| ***Reentrancy*** | Non Reentrant | |
| ***Parameters (in)*** | CtrlIdx | Index of the controller within the context of the Driver |
| ***Parameters (inout)*** | None | |
| ***Parameters (out)*** | TxErrorCounterValues | List of values to read statistic error counter values for transmission. |
| ***Return value*** | Std_ReturnType | `E_OK`: success,<br>`E_NOTOK`: Tx-statistics could not be obtained |

▽

△

| | |
|---|---|
| ***Description*** | Returns the list of Transmission Error Counters out of IETF RFC1213 and RFC1643 defined with Eth_TxErrorCounterValuesType, where the maximal possible value shall denote an invalid value, e.g. this counter is not available. |
| ***Available via*** | CanXL.h |

⌋

Note: This API is derived from Ethernet Driver ([SWS_Eth_91006]). For better understanding of the API's original intention you may check [8, Ethernet Interface] and [11, Ethernet Driver]. It is called by the AUTOSAR Ethernet Interface to achieve compatible CAN XL agnostic behavior.

**[CP_SWS_CanXL_00021]** ⌈The service `CanXL_GetTxErrorCounterValues()` has no functionality and shall always return E_NOT_OK without performing any action.⌋

### 8.3.10 CanXL_GetTxStats

**[CP_SWS_CanXL_10011] Definition of API function CanXL_GetTxStats** ⌈

| | | |
|---|---|---|
| ***Service Name*** | CanXL_GetTxStats | |
| ***Syntax*** | `Std_ReturnType CanXL_GetTxStats ( uint8 CtrlIdx, Eth_TxStatsType* TxStats )` | |
| ***Service ID [hex]*** | 0x1c | |
| ***Sync/Async*** | Synchronous | |
| ***Reentrancy*** | Non Reentrant | |
| ***Parameters (in)*** | CtrlIdx | Index of the controller within the context of the Driver |
| ***Parameters (inout)*** | None | |
| ***Parameters (out)*** | TxStats | List of values to read statistic values for transmission. |
| ***Return value*** | Std_ReturnType | `E_OK`: success,<br>`E_NOTOK`: Tx-statistics could not be obtained |
| ***Description*** | Returns the list of Transmission Statistics out of IETF RFC1213 defined with Eth_TxStatsType, where the maximal possible value shall denote an invalid value, e.g. this counter is not available. | |
| ***Available via*** | CanXL.h | |

⌋

Note: This API is derived from Ethernet Driver ([SWS_Eth_91005]). For better understanding of the API's original intention you may check [8, Ethernet Interface] and [11, Ethernet Driver]. It is called by the AUTOSAR Ethernet Interface to achieve compatible CAN XL agnostic behavior.

**[CP_SWS_CanXL_00022]** ⌈The service `CanXL_GetTxStats()` has no functionality and shall always return E_NOT_OK without performing any action.⌋

### 8.3.11 CanXL_ProvideTxBuffer

**[CP_SWS_CanXL_10012] Definition of API function CanXL_ProvideTxBuffer** ⌈

| | | |
|---|---|---|
| ***Service Name*** | CanXL_ProvideTxBuffer | |
| ***Syntax*** | `BufReq_ReturnType CanXL_ProvideTxBuffer ( uint8 CtrlIdx, uint8 QueueIdx, Eth_BufIdxType* BufIdxPtr, uint8** BufPtr, uint16* LenBytePtr )` | |
| ***Service ID [hex]*** | 0x24 | |
| ***Sync/Async*** | Synchronous | |
| ***Reentrancy*** | Reentrant for different CtrlIdxs and/or QueueIdxs. Non reentrant for the same CtrlIdx and QueueIdx. | |
| ***Parameters (in)*** | CtrlIdx | Index of the controller within the context of the Driver |
| | QueueIdx | Index of the egress queue within the context of the Driver |
| ***Parameters (inout)*** | LenBytePtr | In: desired length in bytes, out: granted length in bytes |
| ***Parameters (out)*** | BufIdxPtr | Index to the granted buffer resource. To be used for subsequent requests |
| | BufPtr | Pointer to the granted buffer |
| ***Return value*** | BufReq_ReturnType | `BUFREQ_OK`: success<br>`BUFREQ_E_NOT_OK`: request not accepted.<br>`BUFREQ_E_BUSY`: all buffers in use<br>`BUFREQ_E_OVFL`: requested buffer too large |
| ***Description*** | Provides access to a transmit buffer of the indicated queue. | |
| ***Available via*** | CanXL.h | |

⌋

Note: This API is derived from Ethernet Driver ([SWS_Eth_00077]). For better understanding of the API's original intention you may check [8, Ethernet Interface] and [11, Ethernet Driver].

**[CP_SWS_CanXL_00058]** ⌈The function shall provide a transmit buffer resource. The `CAN XL Driver` shall lock the buffer until it receives a subsequent call of `CanXL_Transmit()` service with the buffer index returned in the BufIdxPtr parameter.⌋

**[CP_SWS_CanXL_00059]** ⌈From the `CanXLEthEgressFifoIdx` of the controller that matches the QueueIdx, the contained parameter `CanXLEthEgressFifoCanXLPriority` and `CanXLEthEgressFifoCanXLQueue` shall be used.⌋

**[CP_SWS_CanXL_00139] Value range of the returned buffer index**

*Status:* DRAFT
*Upstream requirements:* SRS_Eth_00188

⌈The returned buffer index value of type `Eth_BufIdxType` shall be greater than $2^{16}$-1. The value range for the buffer index shall be:

- 0x00 01 00 00 ... 0xFF FF FF FF: valid
- 0x00 00 00 00 ... 0x00 00 FF FF: reserved for `TxHandleId` of `CanXL_ImmediateTransmit`

⌋

Note: Constraining the buffer index is needed, since TxHandleId of CanXL_ImmediateTransmit used for direct data provision (used as PDU-ID) and BufIdxPtr of CanXL_ProvideTxBuffer used for indirect data provision could overlap. EthIf need an unambiguous id (non-overlapping value range) that corresponds to a transmission request, to idenfy the affected tranmssion request for transmission confirmation via EthIf_TxConfirmation.

**[CP_SWS_CanXL_00060]** ⌈If a buffer requested with CanXL_ProvideTxBuffer() that is larger than the available buffer length, the buffer shall not be locked but return the available length and BUFREQ_E_OVFL.⌋

**[CP_SWS_CanXL_00061]** ⌈If all available buffers are in use the component shall return BUFREQ_E_BUSY.⌋

**[CP_SWS_CanXL_00063]** ⌈If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error CANXL_E_PARAM_CONTROLLER.⌋

**[CP_SWS_CanXL_00142]** ⌈If development error detection is enabled: the function shall check the parameter QueueIdx for being valid. If the check fails, the function shall raise the development error CANXL_E_INV_PARAM.⌋

**[CP_SWS_CanXL_00064]** ⌈If development error detection is enabled: the function shall check the parameter BufIdxPtr for being valid. If the check fails, the function shall raise the development error CANXL_E_PARAM_POINTER.⌋

**[CP_SWS_CanXL_00065]** ⌈If development error detection is enabled: the function shall check the parameter BufPtr for being valid. If the check fails, the function shall raise the development error CANXL_E_PARAM_POINTER.⌋

**[CP_SWS_CanXL_00066]** ⌈If development error detection is enabled: the function shall check the parameter LenBytePtr for being valid. If the check fails, the function shall raise the development error CANXL_E_PARAM_POINTER.⌋

**[CP_SWS_CanXL_00067]** ⌈If development error detection is enabled: CanXL_ProvideTxBuffer() shall raise the error CANXL_E_UNINIT if the driver is not yet initialized.⌋

### 8.3.12 CanXL_ReleaseRxBuffer

**[SWS_CanXL_91000] Definition of API function CanXL_ReleaseRxBuffer**

*Status:* DRAFT
*Upstream requirements:* SRS_Eth_00172

⌈

| | | |
|---|---|---|
| ***Service Name*** | CanXL_ReleaseRxBuffer (draft) | |
| ***Syntax*** | `void CanXL_ReleaseRxBuffer ( uint8 CtrlIdx, Eth_BufIdxType RxHandleId )` | |
| ***Service ID [hex]*** | 0x27 | |
| ***Sync/Async*** | Synchronous | |
| ***Reentrancy*** | Reentrant for different Rx handle ids and Ctrl indexes | |
| ***Parameters (in)*** | CtrlIdx | Index of the controller within the context of the Driver |
| | RxHandleId | Unique receive handle id provided by the Ethernet Driver in a previous call of EthIf_RxIndication, to identify the ingress queue element per physical Ethernet controller |
| ***Parameters (inout)*** | None | |
| ***Parameters (out)*** | None | |
| ***Return value*** | None | |
| ***Description*** | Indication from the upper layer to release the reception buffer (ingress queue element) of the given physical Ethernet controller.<br>**Tags:** atp.Status=draft | |
| ***Available via*** | CanXL.h | |

⌋

Note: `CanXL_ReleaseRxBuffer()` could be called in context of an `EthIf_RxIndication()` call

### 8.3.13 CanXL_SetControllerMode

**[CP_SWS_CanXL_10016] Definition of API function CanXL_SetControllerMode** ⌈

| | | |
|---|---|---|
| ***Service Name*** | CanXL_SetControllerMode | |
| ***Syntax*** | `Std_ReturnType CanXL_SetControllerMode ( uint8 CtrlIdx, Eth_ModeType CtrlMode )` | |
| ***Service ID [hex]*** | 0x1a | |
| ***Sync/Async*** | Asynchronous | |
| ***Reentrancy*** | Non Reentrant | |
| ***Parameters (in)*** | CtrlIdx | Index of the controller within the context of the Driver |
| | CtrlMode | ETH_MODE_DOWN: Disable Rx/Tx communication of the controller<br>ETH_MODE_ACTIVE: Enable Rx/Tx communication of the controller |
| ***Parameters (inout)*** | None | |

▽

△

| | | |
|---|---|---|
| ***Parameters (out)*** | None | |
| ***Return value*** | Std_ReturnType | `E_OK`: Service accepted<br>`E_NOT_OK`: Service denied |
| ***Description*** | Enables / Disables Rx/Tx communication of the indexed controller. The result is reported asynchronously via EthIf_CtrlModeIndication. | |
| ***Available via*** | CanXL.h | |

⌋

Note: This API is derived from Ethernet Driver ([SWS_Eth_91009]). For better understanding of the API's original intention you may check [8, Ethernet Interface] and [11, Ethernet Driver].

**[CP_SWS_CanXL_00023]** ⌈The function shall store the current communication state of the indexed controller without influencing the CAN XL controller hardware.⌋

See also Chapter 7.2.1.

**[CP_SWS_CanXL_00024]** ⌈If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error `CANXL_E_PARAM_CONTROLLER`.⌋

**[CP_SWS_CanXL_00130]** ⌈If development error detection is enabled: `CanXL_SetControllerMode()` shall raise the error `CANXL_E_UNINIT` if the driver is not yet initialized.⌋

### 8.3.14 CanXL_SetPhysAddr

**[CP_SWS_CanXL_10015] Definition of API function CanXL_SetPhysAddr** ⌈

| | | |
|---|---|---|
| ***Service Name*** | CanXL_SetPhysAddr | |
| ***Syntax*** | `void CanXL_SetPhysAddr ( uint8 CtrlIdx, const uint8* PhysAddrPtr )` | |
| ***Service ID [hex]*** | 0x13 | |
| ***Sync/Async*** | Synchronous | |
| ***Reentrancy*** | Non Reentrant for the same CtrlIdx, reentrant for different | |
| ***Parameters (in)*** | CtrlIdx | Index of the controller within the context of the Driver. |
| | PhysAddrPtr | Pointer to memory containing the physical source address (MAC address) in network byte order. |
| ***Parameters (inout)*** | None | |
| ***Parameters (out)*** | None | |
| ***Return value*** | None | |
| ***Description*** | Sets the physical source address used by the indexed controller | |
| ***Available via*** | CanXL.h | |

⌋

Note: This API is derived from Ethernet Driver ([SWS_Eth_00151]). For better understanding of the API's original intention you may check [8, Ethernet Interface] and [11, Ethernet Driver].

**[CP_SWS_CanXL_00073]** ⌈The function shall update the source address used by the indexed controller (see `CanXLEthPhysAddress`).⌋

**[CP_SWS_CanXL_00075]** ⌈If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error `CANXL_E_PARAM_CONTROLLER`.⌋

**[CP_SWS_CanXL_00076]** ⌈If development error detection is enabled: the function shall check the parameter PhysAddrPtr for being valid. If the check fails, the function shall raise the development error `CANXL_E_PARAM_POINTER`.⌋

**[CP_SWS_CanXL_00077]** ⌈If development error detection is enabled: `CanXL_SetPhysAddr()` shall raise the error `CANXL_E_UNINIT` if the driver is not yet initialized.⌋

### 8.3.15 CanXL_Transmit

**[CP_SWS_CanXL_10003] Definition of API function CanXL_Transmit** ⌈

| | | |
|---|---|---|
| ***Service Name*** | CanXL_Transmit | |
| ***Syntax*** | `Std_ReturnType CanXL_Transmit ( uint8 CtrlIdx, Eth_BufIdxType BufIdx, Eth_FrameType FrameType, boolean TxConfirmation, uint16 LenByte, const uint8* PhysAddrPtr )` | |
| ***Service ID [hex]*** | 0x1e | |
| ***Sync/Async*** | Synchronous | |
| ***Reentrancy*** | Reentrant for different CtrlIdxs and/or BufIdxs. Non reentrant for the same CtrlIdx and BufIdx. | |
| ***Parameters (in)*** | CtrlIdx | Index of the controller within the context of the Driver |
| | BufIdx | Index of the buffer resource |
| | FrameType | Ethernet frame type |
| | TxConfirmation | Activates transmission confirmation |
| | LenByte | Data length in byte |
| | PhysAddrPtr | Physical target address (MAC address) in network byte order |
| ***Parameters (inout)*** | None | |
| ***Parameters (out)*** | None | |
| ***Return value*** | Std_ReturnType | `E_OK`: success<br>`E_NOT_OK`: transmission failed |
| ***Description*** | Triggers transmission of a previously filled transmit buffer | |
| ***Available via*** | CanXL.h | |

⌋

Note: This API is derived from Ethernet Driver ([SWS_Eth_00087]). For better understanding of the API's original intention you may check [8, Ethernet Interface] and [11, Ethernet Driver].

**[CP_SWS_CanXL_00078]** ⌈The function shall build the Ethernet header with the given physical target address (MAC address) and trigger the transmission of a previously filled transmit buffer.⌋

After transmission, the driver needs to release the allocated buffer. It is up to the implementation when the actual buffer release shall occur, e.g. within the context of the Can_MainFunction, or during the next CanXL_ProvideTxBuffer.

Note: Each successful transmission results in a `SDU Type` 05h (mapped tunneled 802.3 Ethernet frames) `XLFF` on the `CAN XL` pyhsical bus.

**[CP_SWS_CanXL_00081]** ⌈If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error `CANXL_E_PARAM_CONTROLLER`.⌋

**[CP_SWS_CanXL_00082]** ⌈If development error detection is enabled: the function shall check the parameter BufIdx for being valid. If the check fails, the function shall raise the development error `CANXL_E_INV_PARAM`.⌋

**[CP_SWS_CanXL_00083]** ⌈If development error detection is enabled: the function shall check the parameter PhysAddrPtr for being valid. If the check fails, the function shall raise the development error `CANXL_E_PARAM_POINTER`.⌋

**[CP_SWS_CanXL_00084]** ⌈If development error detection is enabled: the function shall check the controller mode for being active. If the check fails, the function shall raise the development error `CANXL_E_INV_MODE`.⌋

**[CP_SWS_CanXL_00085]** ⌈`CanXL_Transmit()` shall return `E_NOT_OK` if it is called withour a prior call to `CanXL_ProvideTxBuffer()`.⌋

**[CP_SWS_CanXL_00131]** ⌈If development error detection is enabled: `CanXL_Transmit()` shall raise the error `CANXL_E_UNINIT` if the driver is not yet initialized.⌋

### 8.3.16 CanXL_ImmediateTransmit

**[SWS_CanXL_91002] Definition of API function CanXL_ImmediateTransmit**

*Status:* DRAFT
*Upstream requirements:* SRS_Eth_00173

⌈

| | | |
|---|---|---|
| ***Service Name*** | CanXL_ImmediateTransmit (draft) | |
| ***Syntax*** | `Std_ReturnType CanXL_ImmediateTransmit (`<br>`  uint8 CtrlIdx,`<br>`  uint8 QueueIdx,`<br>`  Eth_BufIdxType TxHandleId,`<br>`  ListElemStructType* HeaderListPtr,`<br>`  uint8* PayloadPtr,`<br>`  uint16 PayloadLength,`<br>`  boolean AsyncBuffers`<br>`)` | |
| ***Service ID [hex]*** | 0x26 | |
| ***Sync/Async*** | Synchronous | |
| ***Reentrancy*** | Reentrant for different CtrlIdxs and/or QueueIdxs. Non reentrant for the same CtrlIdx and QueueIdx. | |
| ***Parameters (in)*** | CtrlIdx | Index of the controller within the context of the Driver |
| | QueueIdx | Index of the egress queue within the context of the Driver |
| | TxHandleId | Unique transmit handle id provided by the Ethernet Interface, to identify the transmission request per physical Ethernet controller |
| | HeaderListPtr | Pointer to first Ethernet frame header of a single linked list. |
| | PayloadPtr | Pointer to the payload of the Ethernet frame |
| | PayloadLength | Length of the payload |
| | AsyncBuffers | True: Payload and header buffers stay valid until EthIf_Tx Confirmation is called.<br>False: Payload and header buffers are released after return from this function. |
| ***Parameters (inout)*** | None | |
| ***Parameters (out)*** | None | |
| ***Return value*** | Std_ReturnType | `E_OK`: Transmit request has been accepted.<br>`E_NOT_OK`:Transmit request has been rejected. |
| ***Description*** | Request transmission of an Ethernet frame, where each upper layer a header part as element of a single linked list. All headers together with the payload form an entire Ethernet frame<br>**Tags:** atp.Status=draft | |
| ***Available via*** | CanXL.h | |

⌋

### 8.3.17 CanXL_UpdatePhysAddrFilter

**[CP_SWS_CanXL_10013] Definition of API function CanXL_UpdatePhysAddrFilter** ⌈

| | | |
|---|---|---|
| ***Service Name*** | CanXL_UpdatePhysAddrFilter | |
| ***Syntax*** | `Std_ReturnType CanXL_UpdatePhysAddrFilter ( uint8 CtrlIdx, const uint8* PhysAddrPtr, uint16 VlanId, Eth_FilterActionType Action )` | |
| ***Service ID [hex]*** | 0x23 | |
| ***Sync/Async*** | Synchronous | |
| ***Reentrancy*** | Non Reentrant for the same CtrlIdx, reentrant for different | |
| ***Parameters (in)*** | CtrlIdx | Index of the controller within the context of the Driver |
| | PhysAddrPtr | Pointer to memory containing the physical destination address (MAC address) in network byte order. This is the multicast destination address of the layer 2 packet. |
| | VlanId | Identifier of virtual LAN |
| | Action | Add or remove the address from the controller's multicast filter. |
| ***Parameters (inout)*** | None | |
| ***Parameters (out)*** | None | |
| ***Return value*** | Std_ReturnType | `E_OK`: filter was successfully changed<br>`E_NOT_OK`: filter could not be changed |
| ***Description*** | Update the multicast filter of the indexed controller. If the controller is not capable to do the filtering, the software has to do this. In addition, the function is responsible for managing promiscuous mode. | |
| ***Available via*** | CanXL.h | |

⌋

Note: This API is derived from Ethernet Driver ([SWS_Eth_00152]). For better understanding of the API's original intention you may check [8, Ethernet Interface] and [11, Ethernet Driver].

**[CP_SWS_CanXL_00092]** ⌈The function shall update the multicast physical address and `VlanId` receive filter of the indexed controller in case it is available. If parameter `VlanId` is set to 0xFFFF, the Vlan filtering is not used.⌋

Note for [CP_SWS_CanXL_00092]: See Chapter 7.3 "Reception Handling".

**[CP_SWS_CanXL_00096]** ⌈If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error `CANXL_E_PARAM_CONTROLLER`.⌋

**[CP_SWS_CanXL_00097]** ⌈If development error detection is enabled the function shall check the parameter PhysAddrPtr for being valid. If the check fails, the function shall raise the development error `CANXL_E_PARAM_POINTER`.⌋

**[CP_SWS_CanXL_00098]** ⌈If development error detection is enabled: `CanXL_UpdatePhysAddrFilter()` shall raise the error `CANXL_E_UNINIT` if the driver is not yet initialized.⌋

### 8.3.18 CanXL_Write

**[CP_SWS_CanXL_10002] Definition of API function CanXL_Write** ⌈

| | | |
|---|---|---|
| ***Service Name*** | CanXL_Write | |
| ***Syntax*** | `Std_ReturnType CanXL_Write ( Can_HwHandleType Hth, const CanXL_PduType* PduInfo, boolean AsyncBuffer )` | |
| ***Service ID [hex]*** | 0x29 | |
| ***Sync/Async*** | Synchronous | |
| ***Reentrancy*** | Reentrant for different Hth | |
| ***Parameters (in)*** | 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. |
| | PduInfo | Pointer to SDU user memory, Data Length and Identifier. |
| | AsyncBuffer | True: The PDU buffer stays valid until CanIf_TxConfirmation is called.<br>False: The PDU buffer is released after return from this function. |
| ***Parameters (inout)*** | None | |
| ***Parameters (out)*** | None | |
| ***Return value*** | Std_ReturnType | `E_OK`: Write command has been accepted<br>`E_NOT_OK`: development error occurred<br>`CAN_BUSY`: No TX hardware buffer available or pre-emptive call of CanXL_Write that can't be implemented re-entrant (see Can_ReturnType) |
| ***Description*** | This function is called by CanIf to pass a CAN XL message to the CAN XL driver for transmission. It provides the CAN XL specific parameters besides the hardware handle and actual PDU information. | |
| ***Available via*** | CanXL.h | |

⌋

The function is the counterpart to `Can_Write()` and expects additional `XLFF` specific parameters given by XLParams in the PduInfo.

While `Can_Write()` is used to request the transmission of CAN 2.0/FD frames, this function is used to request the transmission of CAN XL frames with following `SDU Types`:

- 01h (content based CAN XL frames)
- 03h (tunneled CAN 2.0/FD frames)
- further SDU Types defined in [2, CiA611-1]

SDU Type 05h (mapped tunneled 802.3 Ethernet frames) is not supported by this function and is exclusively used by `CanXL_Transmit()`.

For CAN 2.0 and CAN FD frames, a CanHardwareObject is configured as CAN HTH, while for CAN XL frames a CanXLHardwareObject is configured as CAN XL HTHs. A CanHardwareObject and a CanXLHardwareObject may share the same hardware queue.

**[CP_SWS_CanXL_00121]** ⌈When `CanXL_Write()` is called for transmitting a tunneled CAN 2.0/FD frame (`SDU Type` 03h), it is responsible to prepare the `LLC data`. Refer to [2, CiA611-1] for the structure. The `LLC data byte 1` shall be assembled from:

- ESI is always error active
- BRS is always disabled
- DLC of tunneled CAN 2.0/FD frame

⌋

For other `SDU Types` the `SDU` data is transferred directly to hardware.

**[CP_SWS_CanXL_00126]** ⌈`CanXL_Write()` shall accept a null pointer as SDU ( `Can_PduType.Can_SduPtrType = NULL`) if the trigger transmit API is enabled for this hardware object (`CanTriggerTransmitEnable = TRUE`).⌋

**[CP_SWS_CanXL_00127]** ⌈If the trigger transmit API is enabled for the hardware object, `CanXL_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, `CanXL_Write()` shall call `CanIf_TriggerTransmit()` with the total size of the allocated message buffer to acquire the PDU's data.⌋

The function first checks if the hardware transmit object that is identified by the HTH is free and if another transmission request is ongoing for the same HTH.

**[CP_SWS_CanXL_00103]** ⌈The function shall perform no actions if the hardware transmit object is busy with another transmit request for an L-PDU and shall return CAN_BUSY.⌋

**[CP_SWS_CanXL_00104]**

*Upstream requirements:* SRS_BSW_00312

⌈The function shall return CAN_BUSY if a preemptive call of `CanXL_Write()` has been issued, that could not be handled reentrant (i.e. a call with the same HTH).⌋

**[CP_SWS_CanXL_00106]**

*Upstream requirements:* SRS_BSW_00369

⌈If development error detection is enabled: `CanXL_Write()` shall raise the error `CANXL_E_UNINIT` if the driver is not yet initialized.⌋

**[CP_SWS_CanXL_00107]**

*Upstream requirements:* SRS_BSW_00323, SRS_BSW_00369

⌈If development error detection is enabled: The function shall raise the error `CANXL_E_PARAM_HANDLE` if the parameter Hth is not a configured CAN XL Hardware Transmit Handle.⌋

**[CP_SWS_CanXL_00109]**

*Upstream requirements:* SRS_BSW_00323, SRS_BSW_00369

⌈If development error detection is enabled: The function shall raise the error CANXL_E_INV_PARAM if the given SduType has the value 05h.⌋

**[CP_SWS_CanXL_00108]**

*Upstream requirements:* SRS_BSW_00323, SRS_BSW_00369

⌈If development error detection is enabled: If SDU Type has another value than 03h, the function shall raise the error CANXL_E_PARAM_DATA_LENGTH if the length is 0 or exceeding 2048.⌋

**[CP_SWS_CanXL_00116]**

*Upstream requirements:* SRS_BSW_00323, SRS_BSW_00369

⌈If development error detection is enabled: The function shall raise the error CANXL_E_INV_PARAM if the PduInfo is inconsistent according to CiA 611-1 SDU types chapter 5.⌋

**[CP_SWS_CanXL_00119]**

*Upstream requirements:* SRS_BSW_00323, SRS_BSW_00369

⌈If development error detection is enabled: The function shall raise the error CANXL_E_INV_PARAM if PduInfo->XLParams->Vcid is larger than 255.⌋

**[CP_SWS_CanXL_00110]**

*Upstream requirements:* SRS_BSW_00323, SRS_BSW_00369

⌈If development error detection is enabled: The function shall raise CANXL_E_PARAM_POINTER if the parameter PduInfo is a null pointer.⌋

**[CP_SWS_CanXL_00112]**

*Upstream requirements:* SRS_BSW_00323, SRS_BSW_00369

⌈If development error detection is enabled: The function shall raise CANXL_E_PARAM_POINTER if the XLParams pointer is a null pointer.⌋

### 8.3.19 CanXL_XLReleaseRxBuffer

**[SWS_CanXL_91003] Definition of API function CanXL_XLReleaseRxBuffer ⌈**

| | |
|---|---|
| ***Service Name*** | CanXL_XLReleaseRxBuffer |
| ***Syntax*** | `void CanXL_XLReleaseRxBuffer (`<br>`  Can_HwHandleType Hrh`<br>`)` |
| ***Service ID [hex]*** | 0x2a |

▽

△

| | | |
|---|---|---|
| ***Sync/Async*** | Synchronous | |
| ***Reentrancy*** | Reentrant for different Hrh | |
| ***Parameters (in)*** | Hrh | Information which HW-transmit handle was used for the reception. Implicitly this is also the information about the used controller because the Hrh numbers are unique inside one hardware unit. |
| ***Parameters (inout)*** | None | |
| ***Parameters (out)*** | None | |
| ***Return value*** | None | |
| ***Description*** | Indication from CanIf to release the CAN XL reception buffer. | |
| ***Available via*** | CanXL.h | |

⌋

## 8.4 Scheduled functions

Note: `CAN XL Driver` does not have additional scheduled functions.

## 8.5 Expected interfaces

In this chapter all interfaces required from other modules are listed.

### 8.5.1 Mandatory interfaces

Note: This section defines all interfaces, which are required to fulfill the core functionality of the module.

**[CP_SWS_CanXL_10022] Definition of mandatory interfaces required by module CanXL** ⌈

| *API Function* | *Header File* | *Description* |
|---|---|---|
| CanIf_XLReportErrorState | CanIf.h | Reports each change of the CAN error state. |
| CanIf_XLRxIndication | CanIf.h | This service indicates a successful reception of a received CAN XL Rx L-PDU to the CanIf after passing all filters and validation checks. It provides the CAN XL specific parameters besides the hardware and actual L-PDU information. |

⌋

### 8.5.2 Optional interfaces

Note: This section defines all interfaces, which are required to fulfill an optional functionality of the module.

**[CP_SWS_CanXL_10021] Definition of optional interfaces requested by module CanXL** ⌈

| *API Function* | *Header File* | *Description* |
|---|---|---|
| EthIf_CtrlModeIndication | EthIf.h | Called asynchronously when mode has been read out. Triggered by previous <EthDrv>_SetController Mode call. Can directly be called within the trigger functions. |
| EthIf_RxIndication | EthIf.h | Receive indication of an Ethernet frame which was received by the indexed controller |
| EthIf_TxConfirmation | EthIf.h | Confirms frame transmission by the indexed controller |

⌋

### 8.5.3 Callout interfaces

Note: `CAN XL Driver` does not provide callout interfaces.

### 8.5.4 Configurable interfaces

Note: `CAN XL Driver` does not use configurable interfaces.

# 9 Sequence diagrams

There are no sequence diagrams needed. The sequences do not differ to CAN/Ethernet.

# 10 Configuration specification

In general, this chapter defines configuration parameters and their clustering into containers. In order to support the specification Chapter 10.1 describes fundamentals. It also specifies a template (table) you shall use for the parameter specification. We intend to leave Chapter 10.1 in the specification to guarantee comprehension.

Chapter 10.2 specifies the structure (containers) and the parameters of the module `CAN XL Driver`.

Chapter 10.4 specifies published information of the module `CAN XL Driver`.

## 10.1 How to read this chapter

For details refer to [4] Chapter 10.1 *"Introduction to configuration specification"*.

## 10.2 Containers and configuration parameters

The following chapters summarize all configuration parameters. The detailed meanings of the parameters are described in Chapter 7 and Chapter 8.

### 10.2.1 CanXLGeneral

**[ECUC_Can_00524] Definition of EcucParamConfContainerDef CanXLGeneral** ⌈

| | |
|---|---|
| **Container Name** | CanXLGeneral |
| **Parent Container** | CanGeneral |
| **Upstream mapping defined** | – |
| **Description** | This container is specified in the SWS CAN XL Driver and contains global parameters of the CAN XL Driver. |
| **Multiplicity** | 0..1 |
| **Pre-Compile only** | true |
| **Configuration Parameters** | |

| **Included Parameters** | | |
|---|---|---|
| **Parameter Name** | **Multiplicity** | **ECUC ID** |
| CanXLEthGlobalTimeSupport | 1 | [ECUC_Can_00525] |

| **No Included Containers** |
|---|

⌋

**[ECUC_Can_00525] Definition of EcucBooleanParamDef CanXLEthGlobalTimeSupport** ⌈

| | |
|---|---|
| **Parameter Name** | CanXLEthGlobalTimeSupport |
| **Parent Container** | CanXLGeneral |
| **Upstream mapping defined** | – |
| **Description** | Enables/Disables the Global Time APIs for the Ethernet Interface used when hardware timestamping is supported by CAN controller. |
| **Multiplicity** | 1 |
| **Type** | EcucBooleanParamDef |
| **Default value** | – |
| **Pre-Compile only** | true |

⌋

### 10.2.2 CanXLController

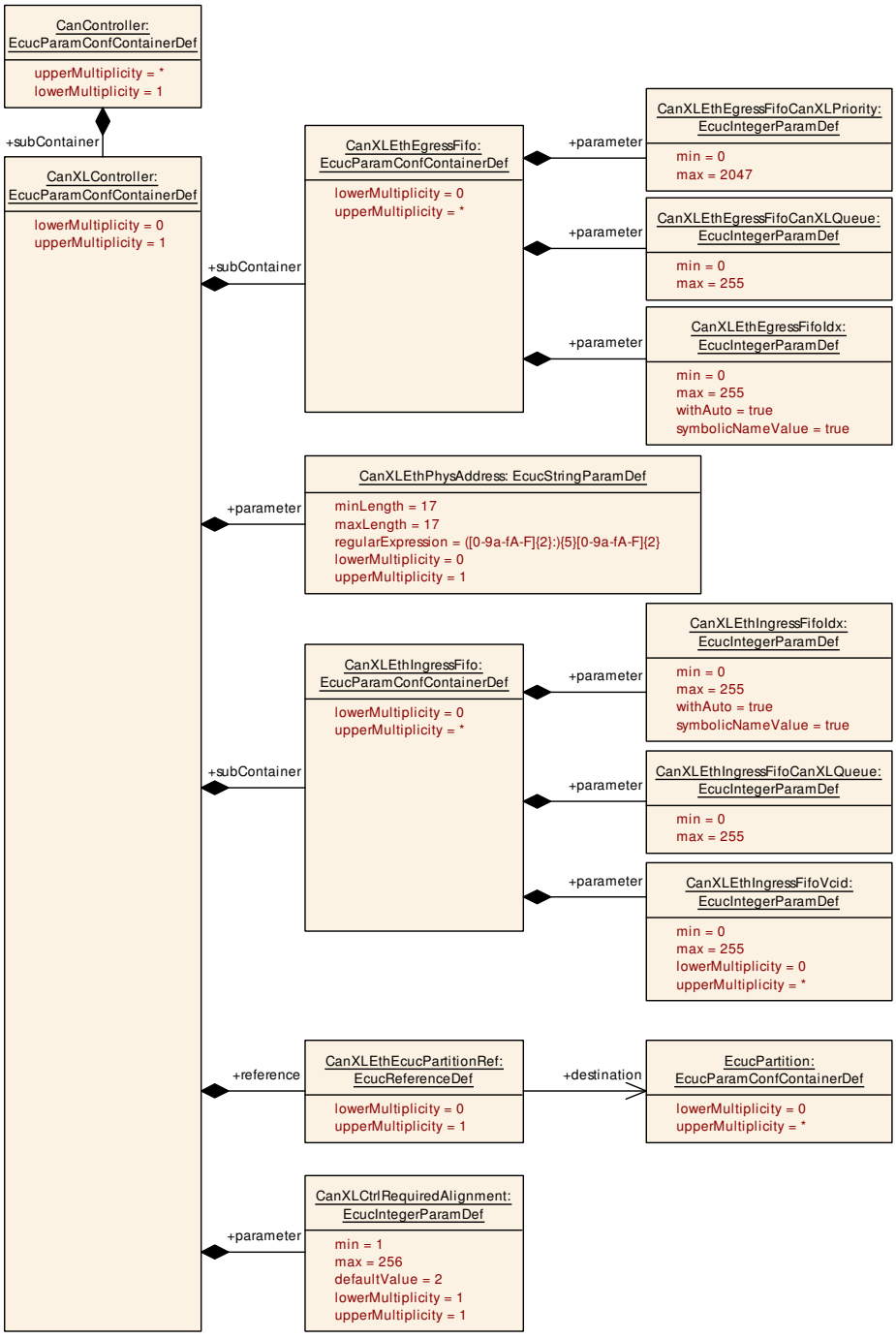


**Figure 10.1: EcucDef CanXLController**

**[ECUC_Can_00499] Definition of EcucParamConfContainerDef CanXLController**
⌈

| Container Name | CanXLController |
|---|---|
| Parent Container | CanController |
| Upstream mapping defined | – |
| Description | This container is specified in the SWS CAN XL Driver and represents a CAN XL channel. If this container is present, the CAN driver will provide the extended CanXL API. |
| Multiplicity | 0..1 |
| Pre-Compile only | true |
| Configuration Parameters | |

| Included Parameters | | |
|---|---|---|
| Parameter Name | Multiplicity | ECUC ID |
| CanXLCtrlRequiredAlignment | 1 | [ECUC_Can_00539] |
| CanXLEthPhysAddress | 0..1 | [ECUC_Can_00506] |
| CanXLEthEcucPartitionRef | 0..1 | [ECUC_Can_00511] |

| Included Containers | | |
|---|---|---|
| Container Name | Multiplicity | Description |
| CanXLCanSecConfig | 0..1 | CANsec configuration for CAN XL |
| CanXLEthEgressFifo | 0..* | Represents a Fifo at the egress side. |
| CanXLEthIngressFifo | 0..* | Represents a Fifo at the ingress side. |

⌋

### [ECUC_Can_00539] Definition of EcucIntegerParamDef CanXLCtrlRequiredAlignment ⌈

| Parameter Name | CanXLCtrlRequiredAlignment | |
|---|---|---|
| Parent Container | CanXLController | |
| Upstream mapping defined | – | |
| Description | Defines the required memory alignment in bytes for source buffers used in hardware supported data transfers. The value of this parameter shall comply with the alignment recommended by the underlying hardware. | |
| Multiplicity | 1 | |
| Type | EcucIntegerParamDef | |
| Range | 1 .. 256 | |
| Default value | 2 | |
| Pre-Compile only | true | |

⌋

### [ECUC_Can_00506] Definition of EcucStringParamDef CanXLEthPhysAddress ⌈

| Parameter Name | CanXLEthPhysAddress |
|---|---|
| Parent Container | CanXLController |
| Upstream mapping defined | – |
| Description | Specifies the unique 48-bit physical address (MAC address) of the controller in network byte order. |
| Multiplicity | 0..1 |

▽

△

| | | |
|---|---|---|
| **Type** | EcucStringParamDef | |
| **Default value** | – | |
| **Length** | 17-17 | |
| **Regular Expression** | ([0-9a-fA-F]{2}:){5}[0-9a-fA-F]{2} | |
| **Post-Build Value** | Loadable | Selectable |

⌋

**[ECUC_Can_00511] Definition of EcucReferenceDef CanXLEthEcucPartitionRef**
⌈

| | |
|---|---|
| **Parameter Name** | CanXLEthEcucPartitionRef |
| **Parent Container** | CanXLController |
| **Upstream mapping defined** | – |
| **Description** | Maps the Ethernet Interface access to the CAN XL controller to zero or one ECUC partitions. |
| **Multiplicity** | 0..1 |
| **Type** | Reference to EcucPartition |
| **Pre-Compile only** | true |

⌋

**[CP_SWS_CanXL_00120]** ⌈The module shall operate as an independent instance in each of the partitions, means the called API will only target the partition it is called in.⌋

**[CP_SWS_CanXL_CONSTR_00001]** ⌈If `CanEcucPartitionRef` references one or more ECUC partitions, `CanXLEthControllerEcucPartitionRef` shall have a multiplicity of one and reference an ECUC partition as well.⌋

Note: `CanXLEthControllerEcucPartitionRef` may reference a different partition than any reference in `CanEcucPartitionRef`.

### 10.2.3 CanXLCanSecConfig

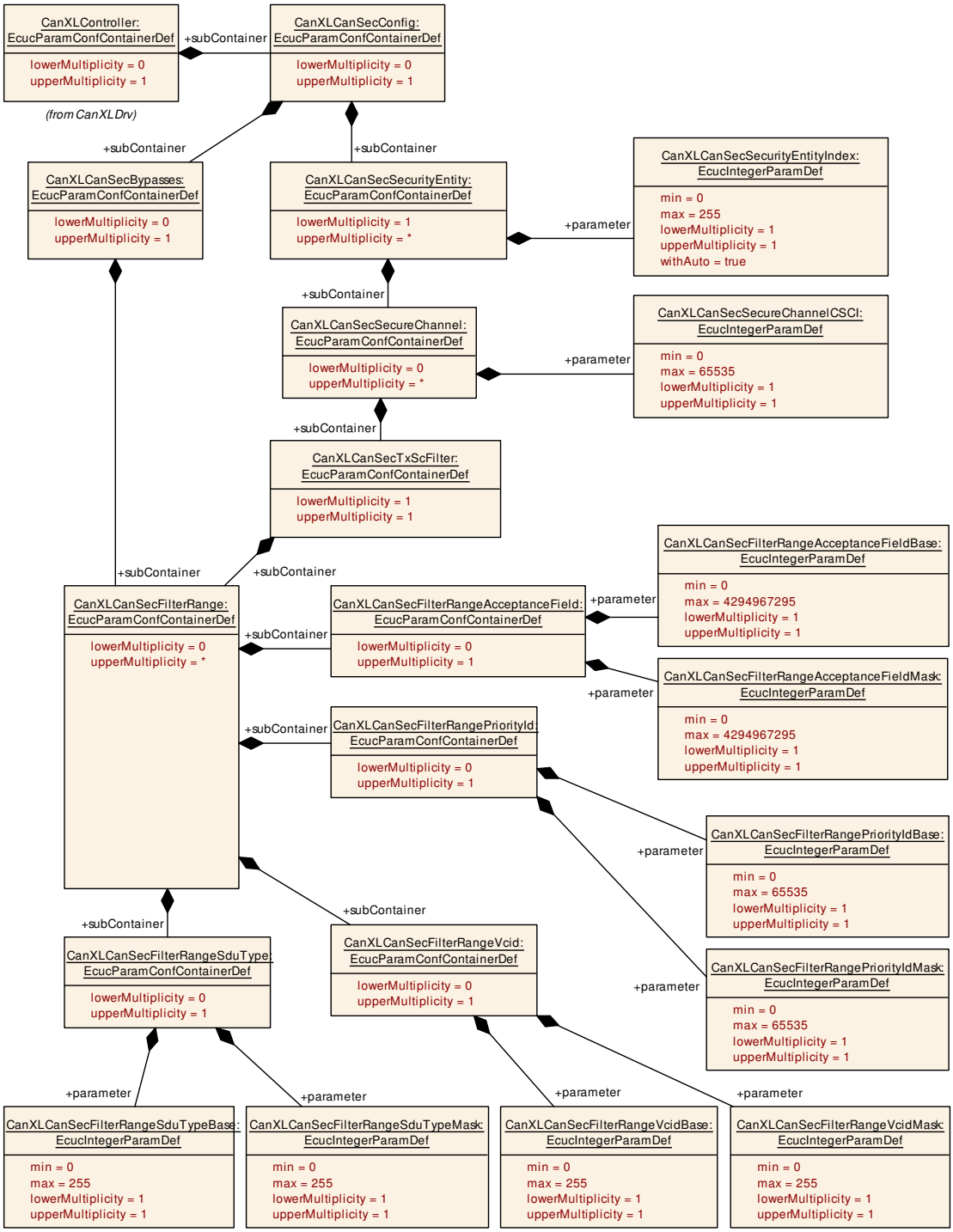


**Figure 10.2: CanXLController CAN sec configuration**

**[ECUC_Can_00540] Definition of EcucParamConfContainerDef CanXLCanSec Config** ⌈

| Container Name | CanXLCanSecConfig | |
|---|---|---|
| Parent Container | CanXLController | |
| Upstream mapping defined | – | |
| Description | CANsec configuration for CAN XL | |
| Multiplicity | 0..1 | |
| Post-Build Multiplicity | Loadable | Selectable |
| Configuration Parameters | | |

| No Included Parameters |
|---|

| Included Containers | | |
|---|---|---|
| Container Name | Multiplicity | Description |
| CanXLCanSecBypasses | 0..1 | Specify elements that should not use CANsec (bypass) |
| CanXLCanSecSecurityEntity | 1..* | Represents a CANsec Security Entity (SecY) |

⌋

### 10.2.3.1 CanXLCanSecBypasses

**[ECUC_Can_00541] Definition of EcucParamConfContainerDef CanXLCanSecBypasses** ⌈

| Container Name | CanXLCanSecBypasses | |
|---|---|---|
| Parent Container | CanXLCanSecConfig | |
| Upstream mapping defined | – | |
| Description | Specify elements that should not use CANsec (bypass) | |
| Multiplicity | 0..1 | |
| Post-Build Multiplicity | Loadable | Selectable |
| Configuration Parameters | | |

| No Included Parameters |
|---|

| Included Containers | | |
|---|---|---|
| Container Name | Multiplicity | Description |
| CanXLCanSecFilterRange | 0..* | Definition of the CanSec Filter Range. |

⌋

### 10.2.3.2 CanXLCanSecSecurityEntity

**[ECUC_Can_00542] Definition of EcucParamConfContainerDef CanXLCanSecSecurityEntity** ⌈

| Container Name | CanXLCanSecSecurityEntity | |
|---|---|---|
| Parent Container | CanXLCanSecConfig | |
| Upstream mapping defined | – | |
| Description | Represents a CANsec Security Entity (SecY) | |
| Multiplicity | 1..* | |
| Post-Build Multiplicity | Loadable | Selectable |
| Configuration Parameters | | |

| Included Parameters | | |
|---|---|---|
| Parameter Name | Multiplicity | ECUC ID |
| CanXLCanSecSecurityEntityIndex | 1 | [ECUC_Can_00543] |

| Included Containers | | |
|---|---|---|
| Container Name | Multiplicity | Description |
| CanXLCanSecSecureChannel | 0..* | Represents a secure channel. Currently, the secure channel is only configured on the Tx side. The Rx side information is managed at runtime by the hardware. |

⌋

### [ECUC_Can_00543] Definition of EcucIntegerParamDef CanXLCanSecSecurity EntityIndex ⌈

| Parameter Name | CanXLCanSecSecurityEntityIndex | |
|---|---|---|
| Parent Container | CanXLCanSecSecurityEntity | |
| Upstream mapping defined | – | |
| Description | Local index for this Security Entity (SecY) in this CanXL controller | |
| Multiplicity | 1 | |
| Type | EcucIntegerParamDef | |
| Range | 0 .. 255 | |
| Default value | – | |
| With Auto | true | |
| Post-Build Value | Loadable | Selectable |

⌋

## 10.2.3.3 CanXLCanSecSecureChannel

### [ECUC_Can_00544] Definition of EcucParamConfContainerDef CanXLCanSec SecureChannel ⌈

| Container Name | CanXLCanSecSecureChannel |
|---|---|
| Parent Container | CanXLCanSecSecurityEntity |
| Upstream mapping defined | – |
| Description | Represents a secure channel. Currently, the secure channel is only configured on the Tx side. The Rx side information is managed at runtime by the hardware. |

▽

△

| | | |
|---|---|---|
| **Multiplicity** | 0..* | |
| **Post-Build Multiplicity** | Loadable | Selectable |
| **Configuration Parameters** | | |

| **Included Parameters** | | |
|---|---|---|
| **Parameter Name** | **Multiplicity** | **ECUC ID** |
| CanXLCanSecSecureChannelCSCI | 1 | [ECUC_Can_00545] |

| **Included Containers** | | |
|---|---|---|
| **Container Name** | **Multiplicity** | **Description** |
| CanXLCanSecTxScFilter | 1 | Each CANsec secure channel identifier (CSCI) has a filter, so that the controller can decide which secure channel (SC) should be used when sending. Currently, the CANsec standard allows 4095 identifiers, but the actual number of filters is hardware dependent. Each driver VSMD should set the upperMultiplicity to match the hardware capabilities. |

⌋

### [ECUC_Can_00545] Definition of EcucIntegerParamDef CanXLCanSecSecureChannelCSCI ⌈

| | | |
|---|---|---|
| **Parameter Name** | CanXLCanSecSecureChannelCSCI | |
| **Parent Container** | CanXLCanSecSecureChannel | |
| **Upstream mapping defined** | – | |
| **Description** | The CANsec secure channel identifier (CSCI) | |
| **Multiplicity** | 1 | |
| **Type** | EcucIntegerParamDef | |
| **Range** | 0 .. 65535 | |
| **Default value** | – | |
| **Post-Build Value** | Loadable | Selectable |

⌋

## 10.2.3.3.1 CanXLCanSecTxScFilter

### [ECUC_Can_00546] Definition of EcucParamConfContainerDef CanXLCanSecTxScFilter ⌈

| | |
|---|---|
| **Container Name** | CanXLCanSecTxScFilter |
| **Parent Container** | CanXLCanSecSecureChannel |
| **Upstream mapping defined** | – |
| **Description** | Each CANsec secure channel identifier (CSCI) has a filter, so that the controller can decide which secure channel (SC) should be used when sending. Currently, the CANsec standard allows 4095 identifiers, but the actual number of filters is hardware dependent. Each driver VSMD should set the upperMultiplicity to match the hardware capabilities. |
| **Multiplicity** | 1 |

▽

△

| Pre-Compile only | true |
|---|---|
| **Configuration Parameters** | |

| No Included Parameters |
|---|

| Included Containers | | |
|---|---|---|
| **Container Name** | **Multiplicity** | **Description** |
| CanXLCanSecFilterRange | 0..* | Definition of the CanSec Filter Range. |

⌋

### 10.2.3.4 CanXLCanSecFilterRange

### [ECUC_Can_00547] Definition of EcucParamConfContainerDef CanXLCanSecFilterRange ⌈

| Container Name | CanXLCanSecFilterRange | |
|---|---|---|
| Parent Container | CanXLCanSecBypasses, CanXLCanSecTxScFilter | |
| Upstream mapping defined | – | |
| Description | Definition of the CanSec Filter Range. | |
| Multiplicity | 0..* | |
| Post-Build Multiplicity | Loadable | Selectable |
| Configuration Parameters | | |

| No Included Parameters |
|---|

| Included Containers | | |
|---|---|---|
| **Container Name** | **Multiplicity** | **Description** |
| CanXLCanSecFilterRange AcceptanceField | 0..1 | Mask for the Acceptance Field (AF) range for this Secure Channel (SC). The mask is defined by CanXLCanSecFilter RangeAcceptanceFieldBase and CanXLCanSecFilterRange AcceptanceFieldMask. |
| CanXLCanSecFilterRangePriority Id | 0..1 | Mask for the Priority ID range for this Secure Channel (SC). The mask is defined by CanXLCanSecFilterRangePriorityIdBase and CanXLCanSecFilterRangePriorityIdMask. |
| CanXLCanSecFilterRangeSdu Type | 0..1 | Mask for the SDU Type range (SDT, SDU = service data unit) for this Secure Channel (SC). The mask is defined by CanXLCan SecFilterRangeSduTypeBase and CanXLCanSecFilterRange SduTypeMask. |
| CanXLCanSecFilterRangeVcid | 0..1 | Mask for the VCID range for this Secure Channel (SC). The mask is defined by CanXLCanSecFilterRangeVcidBase and Can XLCanSecFilterRangeVcidMask. |

⌋

#### 10.2.3.4.1 CanXLCanSecFilterRangeAcceptanceField

**[ECUC_Can_00548] Definition of EcucParamConfContainerDef CanXLCanSecFilterRangeAcceptanceField** ⌈

| | | |
|---|---|---|
| **Container Name** | CanXLCanSecFilterRangeAcceptanceField | |
| **Parent Container** | CanXLCanSecFilterRange | |
| **Upstream mapping defined** | – | |
| **Description** | Mask for the Acceptance Field (AF) range for this Secure Channel (SC). The mask is defined by CanXLCanSecFilterRangeAcceptanceFieldBase and CanXLCanSecFilterRangeAcceptanceFieldMask. | |
| **Multiplicity** | 0..1 | |
| **Post-Build Multiplicity** | Loadable | Selectable |
| **Configuration Parameters** | | |

| **Included Parameters** | | |
|---|---|---|
| **Parameter Name** | **Multiplicity** | **ECUC ID** |
| CanXLCanSecFilterRangeAcceptanceFieldBase | 1 | [ECUC_Can_00550] |
| CanXLCanSecFilterRangeAcceptanceFieldMask | 1 | [ECUC_Can_00551] |

| **No Included Containers** |
|---|

⌋

**[ECUC_Can_00550] Definition of EcucIntegerParamDef CanXLCanSecFilterRangeAcceptanceFieldBase** ⌈

| | | |
|---|---|---|
| **Parameter Name** | CanXLCanSecFilterRangeAcceptanceFieldBase | |
| **Parent Container** | CanXLCanSecFilterRangeAcceptanceField | |
| **Upstream mapping defined** | – | |
| **Description** | Mask for the Acceptance Field (AF) range for this Secure Channel (SC). The mask is defined by CanXLCanSecFilterRangeAcceptanceFieldBase and CanXLCanSecFilterRangeAcceptanceFieldMask. | |
| **Multiplicity** | 1 | |
| **Type** | EcucIntegerParamDef | |
| **Range** | 0 .. 4294967295 | |
| **Default value** | – | |
| **Post-Build Value** | Loadable | Selectable |

⌋

**[ECUC_Can_00551] Definition of EcucIntegerParamDef CanXLCanSecFilterRangeAcceptanceFieldMask** ⌈

| | |
|---|---|
| **Parameter Name** | CanXLCanSecFilterRangeAcceptanceFieldMask |
| **Parent Container** | CanXLCanSecFilterRangeAcceptanceField |
| **Upstream mapping defined** | – |

▽

△

| | | |
|---|---|---|
| **Description** | Mask for the Acceptance Field (AF) range for this Secure Channel (SC). The mask is defined by CanXLCanSecFilterRangeAcceptanceFieldBase and CanXLCanSecFilter RangeAcceptanceFieldMask. | |
| **Multiplicity** | 1 | |
| **Type** | EcucIntegerParamDef | |
| **Range** | 0 .. 4294967295 | |
| **Default value** | – | |
| **Post-Build Value** | Loadable | Selectable |

⌋

### 10.2.3.4.2 CanXLCanSecFilterRangePriorityId

### [ECUC_Can_00549] Definition of EcucParamConfContainerDef CanXLCanSecFilterRangePriorityId ⌈

| | | |
|---|---|---|
| **Container Name** | CanXLCanSecFilterRangePriorityId | |
| **Parent Container** | CanXLCanSecFilterRange | |
| **Upstream mapping defined** | – | |
| **Description** | Mask for the Priority ID range for this Secure Channel (SC). The mask is defined by CanXLCanSecFilterRangePriorityIdBase and CanXLCanSecFilterRangePriorityId Mask. | |
| **Multiplicity** | 0..1 | |
| **Post-Build Multiplicity** | Loadable | Selectable |
| **Configuration Parameters** | | |

| **Included Parameters** | | |
|---|---|---|
| **Parameter Name** | **Multiplicity** | **ECUC ID** |
| CanXLCanSecFilterRangePriorityIdBase | 1 | [ECUC_Can_00552] |
| CanXLCanSecFilterRangePriorityIdMask | 1 | [ECUC_Can_00553] |

| **No Included Containers** |
|---|

⌋

### [ECUC_Can_00552] Definition of EcucIntegerParamDef CanXLCanSecFilterRangePriorityIdBase ⌈

| | | |
|---|---|---|
| **Parameter Name** | CanXLCanSecFilterRangePriorityIdBase | |
| **Parent Container** | CanXLCanSecFilterRangePriorityId | |
| **Upstream mapping defined** | – | |
| **Description** | Mask for the Priority ID range for this Secure Channel (SC). The mask is defined by CanXLCanSecFilterRangePriorityIdBase and CanXLCanSecFilterRangePriorityId Mask. | |
| **Multiplicity** | 1 | |
| **Type** | EcucIntegerParamDef | |
| **Range** | 0 .. 65535 | |

▽

△

| | | |
|---|---|---|
| **Default value** | – | |
| **Post-Build Value** | Loadable | Selectable |

⌋

### [ECUC_Can_00553] Definition of EcucIntegerParamDef CanXLCanSecFilterRangePriorityIdMask ⌈

| | | |
|---|---|---|
| **Parameter Name** | CanXLCanSecFilterRangePriorityIdMask | |
| **Parent Container** | CanXLCanSecFilterRangePriorityId | |
| **Upstream mapping defined** | – | |
| **Description** | Mask for the Priority ID range for this Secure Channel (SC). The mask is defined by CanXLCanSecFilterRangePriorityIdBase and CanXLCanSecFilterRangePriorityId Mask. | |
| **Multiplicity** | 1 | |
| **Type** | EcucIntegerParamDef | |
| **Range** | 0 .. 65535 | |
| **Default value** | – | |
| **Post-Build Value** | Loadable | Selectable |

⌋

## 10.2.3.4.3 CanXLCanSecFilterRangeSduType

### [ECUC_Can_00555] Definition of EcucParamConfContainerDef CanXLCanSecFilterRangeSduType ⌈

| | | |
|---|---|---|
| **Container Name** | CanXLCanSecFilterRangeSduType | |
| **Parent Container** | CanXLCanSecFilterRange | |
| **Upstream mapping defined** | – | |
| **Description** | Mask for the SDU Type range (SDT, SDU = service data unit) for this Secure Channel (SC). The mask is defined by CanXLCanSecFilterRangeSduTypeBase and CanXLCan SecFilterRangeSduTypeMask. | |
| **Multiplicity** | 0..1 | |
| **Post-Build Multiplicity** | Loadable | Selectable |
| **Configuration Parameters** | | |

| **Included Parameters** | | |
|---|---|---|
| **Parameter Name** | **Multiplicity** | **ECUC ID** |
| CanXLCanSecFilterRangeSduTypeBase | 1 | [ECUC_Can_00556] |
| CanXLCanSecFilterRangeSduTypeMask | 1 | [ECUC_Can_00557] |

| **No Included Containers** |
|---|

⌋

### [ECUC_Can_00556] Definition of EcucIntegerParamDef CanXLCanSecFilterRangeSduTypeBase ⌈

| | | |
|---|---|---|
| **Parameter Name** | CanXLCanSecFilterRangeSduTypeBase | |
| **Parent Container** | CanXLCanSecFilterRangeSduType | |
| **Upstream mapping defined** | – | |
| **Description** | Mask for the SDU Type range (SDT, SDU = service data unit) for this Secure Channel (SC). The mask is defined by CanXLCanSecFilterRangeSduTypeBase and CanXLCanSecFilterRangeSduTypeMask. | |
| **Multiplicity** | 1 | |
| **Type** | EcucIntegerParamDef | |
| **Range** | 0 .. 255 | |
| **Default value** | – | |
| **Post-Build Value** | Loadable | Selectable |

⌋

### [ECUC_Can_00557] Definition of EcucIntegerParamDef CanXLCanSecFilterRangeSduTypeMask ⌈

| | | |
|---|---|---|
| **Parameter Name** | CanXLCanSecFilterRangeSduTypeMask | |
| **Parent Container** | CanXLCanSecFilterRangeSduType | |
| **Upstream mapping defined** | – | |
| **Description** | Mask for the SDU Type range (SDT, SDU = service data unit) for this Secure Channel (SC). The mask is defined by CanXLCanSecFilterRangeSduTypeBase and CanXLCanSecFilterRangeSduTypeMask. | |
| **Multiplicity** | 1 | |
| **Type** | EcucIntegerParamDef | |
| **Range** | 0 .. 255 | |
| **Default value** | – | |
| **Post-Build Value** | Loadable | Selectable |

⌋

## 10.2.3.4.4 CanXLCanSecFilterRangeVcid

### [ECUC_Can_00554] Definition of EcucParamConfContainerDef CanXLCanSecFilterRangeVcid ⌈

| | | |
|---|---|---|
| **Container Name** | CanXLCanSecFilterRangeVcid | |
| **Parent Container** | CanXLCanSecFilterRange | |
| **Upstream mapping defined** | – | |
| **Description** | Mask for the VCID range for this Secure Channel (SC). The mask is defined by CanXLCanSecFilterRangeVcidBase and CanXLCanSecFilterRangeVcidMask. | |
| **Multiplicity** | 0..1 | |
| **Post-Build Multiplicity** | Loadable | Selectable |
| **Configuration Parameters** | | |

| Included Parameters | | |
|---|---|---|
| **Parameter Name** | **Multiplicity** | **ECUC ID** |
| CanXLCanSecFilterRangeVcidBase | 1 | [ECUC_Can_00558] |
| CanXLCanSecFilterRangeVcidMask | 1 | [ECUC_Can_00559] |

| No Included Containers |
|---|

⌋

### [ECUC_Can_00558] Definition of EcucIntegerParamDef CanXLCanSecFilterRangeVcidBase ⌈

| Parameter Name | CanXLCanSecFilterRangeVcidBase | |
|---|---|---|
| Parent Container | CanXLCanSecFilterRangeVcid | |
| Upstream mapping defined | – | |
| Description | Mask for the VCID range for this Secure Channel (SC). The mask is defined by CanXLCanSecFilterRangeVcidBase and CanXLCanSecFilterRangeVcidMask. | |
| Multiplicity | 1 | |
| Type | EcucIntegerParamDef | |
| Range | 0 .. 255 | |
| Default value | – | |
| Post-Build Value | Loadable | Selectable |

⌋

### [ECUC_Can_00559] Definition of EcucIntegerParamDef CanXLCanSecFilterRangeVcidMask ⌈

| Parameter Name | CanXLCanSecFilterRangeVcidMask | |
|---|---|---|
| Parent Container | CanXLCanSecFilterRangeVcid | |
| Upstream mapping defined | – | |
| Description | Mask for the VCID range for this Secure Channel (SC). The mask is defined by CanXLCanSecFilterRangeVcidBase and CanXLCanSecFilterRangeVcidMask. | |
| Multiplicity | 1 | |
| Type | EcucIntegerParamDef | |
| Range | 0 .. 255 | |
| Default value | – | |
| Post-Build Value | Loadable | Selectable |

⌋

## 10.2.4 CanXLEthEgressFifo

### [ECUC_Can_00502] Definition of EcucParamConfContainerDef CanXLEthEgressFifo ⌈

| Container Name | CanXLEthEgressFifo | |
|---|---|---|
| Parent Container | CanXLController | |
| Upstream mapping defined | – | |
| Description | Represents a Fifo at the egress side. | |
| Multiplicity | 0..* | |
| Post-Build Multiplicity | Loadable | Selectable |
| Configuration Parameters | | |

| Included Parameters | | |
|---|---|---|
| Parameter Name | Multiplicity | ECUC ID |
| CanXLEthEgressFifoCanXLPriority | 1 | [ECUC_Can_00503] |
| CanXLEthEgressFifoCanXLQueue | 1 | [ECUC_Can_00504] |
| CanXLEthEgressFifoIdx | 1 | [ECUC_Can_00505] |

| No Included Containers |
|---|

⌋

## [ECUC_Can_00503] Definition of EcucIntegerParamDef CanXLEthEgressFifoCanXLPriority ⌈

| Parameter Name | CanXLEthEgressFifoCanXLPriority | |
|---|---|---|
| Parent Container | CanXLEthEgressFifo | |
| Upstream mapping defined | – | |
| Description | Defines the CAN XL Priority ID to be used for outgoing tunneled Ethernet frames using this FIFO. | |
| Multiplicity | 1 | |
| Type | EcucIntegerParamDef | |
| Range | 0 .. 2047 | |
| Default value | – | |
| Post-Build Value | Loadable | Selectable |

⌋

## [ECUC_Can_00504] Definition of EcucIntegerParamDef CanXLEthEgressFifoCanXLQueue ⌈

| Parameter Name | CanXLEthEgressFifoCanXLQueue | |
|---|---|---|
| Parent Container | CanXLEthEgressFifo | |
| Upstream mapping defined | – | |
| Description | Defines the CAN XL Queue to be used for outgoing tunneled Ethernet frames using this FIFO. | |
| Multiplicity | 1 | |
| Type | EcucIntegerParamDef | |
| Range | 0 .. 255 | |
| Default value | – | |
| Post-Build Value | Loadable | Selectable |

⌋

### [ECUC_Can_00505] Definition of EcucIntegerParamDef CanXLEthEgressFifoIdx ⌈

| | | |
|---|---|---|
| **Parameter Name** | CanXLEthEgressFifoIdx | |
| **Parent Container** | CanXLEthEgressFifo | |
| **Upstream mapping defined** | – | |
| **Description** | Egress Fifo index. | |
| **Multiplicity** | 1 | |
| **Type** | EcucIntegerParamDef (Symbolic Name generated for this parameter) | |
| **Range** | 0 .. 255 | |
| **Default value** | – | |
| **With Auto** | true | |
| **Pre-Compile only** | true | |

⌋

## 10.2.5 CanXLEthIngressFifo

### [ECUC_Can_00507] Definition of EcucParamConfContainerDef CanXLEthIngressFifo ⌈

| | | |
|---|---|---|
| **Container Name** | CanXLEthIngressFifo | |
| **Parent Container** | CanXLController | |
| **Upstream mapping defined** | – | |
| **Description** | Represents a Fifo at the ingress side. | |
| **Multiplicity** | 0..* | |
| **Post-Build Multiplicity** | Loadable | Selectable |
| **Configuration Parameters** | | |

| **Included Parameters** | | |
|---|---|---|
| **Parameter Name** | **Multiplicity** | **ECUC ID** |
| CanXLEthIngressFifoCanXLQueue | 1 | [ECUC_Can_00509] |
| CanXLEthIngressFifoIdx | 1 | [ECUC_Can_00508] |
| CanXLEthIngressFifoVcid | 0..* | [ECUC_Can_00510] |

| **No Included Containers** |
|---|

⌋

### [ECUC_Can_00509] Definition of EcucIntegerParamDef CanXLEthIngressFifo CanXLQueue ⌈

| | |
|---|---|
| **Parameter Name** | CanXLEthIngressFifoCanXLQueue |
| **Parent Container** | CanXLEthIngressFifo |
| **Upstream mapping defined** | – |
| **Description** | Defines the CAN XL Queue to be used for incoming tunneled Ethernet frames using this FIFO. |

▽

△

| | | |
|---|---|---|
| **Multiplicity** | 1 | |
| **Type** | EcucIntegerParamDef | |
| **Range** | 0 .. 255 | |
| **Default value** | – | |
| **Post-Build Value** | Loadable | Selectable |

⌋

### [ECUC_Can_00508] Definition of EcucIntegerParamDef CanXLEthIngressFifoIdx ⌈

| | | |
|---|---|---|
| **Parameter Name** | CanXLEthIngressFifoIdx | |
| **Parent Container** | CanXLEthIngressFifo | |
| **Upstream mapping defined** | – | |
| **Description** | Ingress Fifo index. | |
| **Multiplicity** | 1 | |
| **Type** | EcucIntegerParamDef (Symbolic Name generated for this parameter) | |
| **Range** | 0 .. 255 | |
| **Default value** | – | |
| **With Auto** | true | |
| **Pre-Compile only** | true | |

⌋

### [ECUC_Can_00510] Definition of EcucIntegerParamDef CanXLEthIngressFifoVcid ⌈

| | | |
|---|---|---|
| **Parameter Name** | CanXLEthIngressFifoVcid | |
| **Parent Container** | CanXLEthIngressFifo | |
| **Upstream mapping defined** | – | |
| **Description** | Configures a VCID to be accepted by this FIFO. If not present, all VCIDs shall be accepted. | |
| **Multiplicity** | 0..* | |
| **Type** | EcucIntegerParamDef | |
| **Range** | 0 .. 255 | |
| **Default value** | – | |
| **Post-Build Multiplicity** | Loadable | Selectable |
| **Post-Build Value** | Loadable | Selectable |

⌋

## 10.2.6 CanXLBaudrateConfig

### [ECUC_Can_00512] Definition of EcucParamConfContainerDef CanXLBaudrateConfig ⌈

| Container Name | CanXLBaudrateConfig | |
|---|---|---|
| Parent Container | CanControllerBaudrateConfig | |
| Upstream mapping defined | – | |
| Description | This container is specified in the SWS CAN XL Driver and contains bit timing related configuration parameters of the CAN controller(s) for payload and CRC of a CAN XL frame. | |
| Multiplicity | 0..1 | |
| Post-Build Multiplicity | Loadable | Selectable |
| Configuration Parameters | | |

| Included Parameters | | |
|---|---|---|
| **Parameter Name** | **Multiplicity** | **ECUC ID** |
| CanXLBaudRate | 1 | [ECUC_Can_00513] |
| CanXLErrorSignaling | 1 | [ECUC_Can_00523] |
| CanXLPropSeg | 1 | [ECUC_Can_00517] |
| CanXLPwmL | 1 | [ECUC_Can_00514] |
| CanXLPwmO | 1 | [ECUC_Can_00516] |
| CanXLPwmS | 1 | [ECUC_Can_00515] |
| CanXLSeg1 | 1 | [ECUC_Can_00518] |
| CanXLSeg2 | 1 | [ECUC_Can_00519] |
| CanXLSspOffset | 0..1 | [ECUC_Can_00521] |
| CanXLSyncJumpWidth | 1 | [ECUC_Can_00520] |
| CanXLTrcvPwmMode | 1 | [ECUC_Can_00522] |

| No Included Containers |
|---|

⌋

## [ECUC_Can_00513] Definition of EcucFloatParamDef CanXLBaudRate ⌈

| Parameter Name | CanXLBaudRate | |
|---|---|---|
| Parent Container | CanXLBaudrateConfig | |
| Upstream mapping defined | – | |
| Description | Specifies the data segment baud rate of the controller in kbps.<br>Note: The CAN XL baudrate should be at least twice the nominal bitrate so that an error flag can safely destroy a CAN XL frame.<br>Dependency: Has to be at least twice as high as CanControllerBaudRate. | |
| Multiplicity | 1 | |
| Type | EcucFloatParamDef | |
| Range | [0 .. 20000] | |
| Default value | – | |
| Post-Build Value | Loadable | Selectable |

⌋

### [ECUC_Can_00523] Definition of EcucBooleanParamDef CanXLErrorSignaling ⌈

| | | |
|---|---|---|
| **Parameter Name** | CanXLErrorSignaling | |
| **Parent Container** | CanXLBaudrateConfig | |
| **Upstream mapping defined** | – | |
| **Description** | Specifies if error signaling shall be enabled.<br>Dependency: Only relevant if CanXLTrcvPwmMode is disabled. | |
| **Multiplicity** | 1 | |
| **Type** | EcucBooleanParamDef | |
| **Default value** | – | |
| **Post-Build Value** | Loadable | Selectable |

⌋

### [ECUC_Can_00517] Definition of EcucIntegerParamDef CanXLPropSeg ⌈

| | | |
|---|---|---|
| **Parameter Name** | CanXLPropSeg | |
| **Parent Container** | CanXLBaudrateConfig | |
| **Upstream mapping defined** | – | |
| **Description** | Specifies propagation delay in time quantas. | |
| **Multiplicity** | 1 | |
| **Type** | EcucIntegerParamDef | |
| **Range** | 0 .. 255 | |
| **Default value** | – | |
| **Post-Build Value** | Loadable | Selectable |

⌋

### [ECUC_Can_00514] Definition of EcucIntegerParamDef CanXLPwmL ⌈

| | | |
|---|---|---|
| **Parameter Name** | CanXLPwmL | |
| **Parent Container** | CanXLBaudrateConfig | |
| **Upstream mapping defined** | – | |
| **Description** | Specifies the PWM long phase length. | |
| **Multiplicity** | 1 | |
| **Type** | EcucIntegerParamDef | |
| **Range** | 0 .. 255 | |
| **Default value** | – | |
| **Post-Build Value** | Loadable | Selectable |

⌋

### [ECUC_Can_00516] Definition of EcucIntegerParamDef CanXLPwmO ⌈

| | |
|---|---|
| **Parameter Name** | CanXLPwmO |
| **Parent Container** | CanXLBaudrateConfig |
| **Upstream mapping defined** | – |
| **Description** | Specifies the PWM time offset. |
| **Multiplicity** | 1 |
| **Type** | EcucIntegerParamDef |

▽

△

| | | |
|---|---|---|
| **Range** | 0 .. 255 | |
| **Default value** | – | |
| **Post-Build Value** | Loadable | Selectable |

⌋

### [ECUC_Can_00515] Definition of EcucIntegerParamDef CanXLPwmS ⌈

| | | |
|---|---|---|
| **Parameter Name** | CanXLPwmS | |
| **Parent Container** | CanXLBaudrateConfig | |
| **Upstream mapping defined** | – | |
| **Description** | Specifies the PWM short phase length. | |
| **Multiplicity** | 1 | |
| **Type** | EcucIntegerParamDef | |
| **Range** | 0 .. 255 | |
| **Default value** | – | |
| **Post-Build Value** | Loadable | Selectable |

⌋

### [ECUC_Can_00518] Definition of EcucIntegerParamDef CanXLSeg1 ⌈

| | | |
|---|---|---|
| **Parameter Name** | CanXLSeg1 | |
| **Parent Container** | CanXLBaudrateConfig | |
| **Upstream mapping defined** | – | |
| **Description** | Specifies phase segment 1 in time quantas. | |
| **Multiplicity** | 1 | |
| **Type** | EcucIntegerParamDef | |
| **Range** | 0 .. 255 | |
| **Default value** | – | |
| **Post-Build Value** | Loadable | Selectable |

⌋

### [ECUC_Can_00519] Definition of EcucIntegerParamDef CanXLSeg2 ⌈

| | | |
|---|---|---|
| **Parameter Name** | CanXLSeg2 | |
| **Parent Container** | CanXLBaudrateConfig | |
| **Upstream mapping defined** | – | |
| **Description** | Specifies phase segment 2 in time quantas. | |
| **Multiplicity** | 1 | |
| **Type** | EcucIntegerParamDef | |
| **Range** | 0 .. 255 | |
| **Default value** | – | |
| **Post-Build Value** | Loadable | Selectable |

⌋

### [ECUC_Can_00521] Definition of EcucIntegerParamDef CanXLSspOffset ⌈

| | | |
|---|---|---|
| **Parameter Name** | CanXLSspOffset | |
| **Parent Container** | CanXLBaudrateConfig | |
| **Upstream mapping defined** | – | |
| **Description** | Specifies the Transmitter Delay Compensation Offset in minimum time quanta. If this parameter is configured, the Transmitter Delay Compensation is done by measurement of the CAN controller. If not specified, Transmitter Delay Compensation is disabled.<br>See ECUC_Can_00494 for details. | |
| **Multiplicity** | 0..1 | |
| **Type** | EcucIntegerParamDef | |
| **Range** | 0 .. 255 | |
| **Default value** | – | |
| **Post-Build Multiplicity** | Loadable | Selectable |
| **Post-Build Value** | Loadable | Selectable |

⌋

### [ECUC_Can_00520] Definition of EcucIntegerParamDef CanXLSyncJumpWidth ⌈

| | | |
|---|---|---|
| **Parameter Name** | CanXLSyncJumpWidth | |
| **Parent Container** | CanXLBaudrateConfig | |
| **Upstream mapping defined** | – | |
| **Description** | Specifies the synchronization jump width for the controller in time quantas. | |
| **Multiplicity** | 1 | |
| **Type** | EcucIntegerParamDef | |
| **Range** | 0 .. 255 | |
| **Default value** | – | |
| **Post-Build Value** | Loadable | Selectable |

⌋

### [ECUC_Can_00522] Definition of EcucBooleanParamDef CanXLTrcvPwmMode ⌈

| | | |
|---|---|---|
| **Parameter Name** | CanXLTrcvPwmMode | |
| **Parent Container** | CanXLBaudrateConfig | |
| **Upstream mapping defined** | – | |
| **Description** | Specifies if the transceiver shall be set to the PWM mode. | |
| **Multiplicity** | 1 | |
| **Type** | EcucBooleanParamDef | |
| **Default value** | – | |
| **Post-Build Value** | Loadable | Selectable |

⌋

## 10.2.7 CanXLHardwareObject

### [ECUC_Can_00526] Definition of EcucParamConfContainerDef CanXLHardware Object ⌈

| Container Name | CanXLHardwareObject | |
|---|---|---|
| Parent Container | CanConfigSet | |
| Upstream mapping defined | – | |
| Description | This container is specified in the SWS CAN XL Driver and contains the configuration (parameters) of CAN XL Hardware Objects. | |
| Multiplicity | 0..* | |
| Post-Build Multiplicity | Loadable | Selectable |
| Configuration Parameters | | |

| Included Parameters | | |
|---|---|---|
| Parameter Name | Multiplicity | ECUC ID |
| CanObjectType | 1 | [ECUC_Can_00327] |
| CanXLObjectId | 1 | [ECUC_Can_00527] |
| CanControllerRef | 1 | [ECUC_Can_00322] |
| CanMainFunctionRWPeriodRef | 0..1 | [ECUC_Can_00438] |

| Included Containers | | |
|---|---|---|
| Container Name | Multiplicity | Description |
| CanXLHwFilter | 0..* | This container is only valid for CAN XL HRHs and contains the configuration (parameters) of one hardware filter.<br>This container is intentionally left empty, because the parameters are very hardware specific and shall be filled in by the VSMD. |

⌋

For parameter table [ECUC_Can_00327] CanObjectType, see definition below container CanHardwareObject.

### [ECUC_Can_00527] Definition of EcucIntegerParamDef CanXLObjectId ⌈

| Parameter Name | CanXLObjectId | |
|---|---|---|
| Parent Container | CanXLHardwareObject | |
| Upstream mapping defined | – | |
| Description | Holds the handle ID of CAN XL HRH or HTH. | |
| Multiplicity | 1 | |
| Type | EcucIntegerParamDef (Symbolic Name generated for this parameter) | |
| Range | 0 .. 65535 | |
| Default value | – | |
| Pre-Compile only | true | |

⌋

For parameter table [ECUC_Can_00322] CanControllerRef, see definition below container CanHardwareObject.

For parameter table [ECUC_Can_00438] CanMainFunctionRWPeriodRef, see definition below container CanHardwareObject.

### 10.2.8 CanXLHwFilter

**[ECUC_Can_00528] Definition of EcucParamConfContainerDef CanXLHwFilter** ⌈

| | | |
|---|---|---|
| **Container Name** | CanXLHwFilter | |
| **Parent Container** | CanXLHardwareObject | |
| **Upstream mapping defined** | – | |
| **Description** | This container is only valid for CAN XL HRHs and contains the configuration (parameters) of one hardware filter.<br>This container is intentionally left empty, because the parameters are very hardware specific and shall be filled in by the VSMD. | |
| **Multiplicity** | 0..* | |
| **Post-Build Multiplicity** | Loadable | Selectable |
| **Configuration Parameters** | | |

| |
|---|
| **No Included Parameters** |

| |
|---|
| **No Included Containers** |

⌋

## 10.3 Configuration Hints

The `CAN XL` bus is expected to be controlled only via the CAN stack, while the Ethernet stack switches the `CAN XL` bus like a virtual bus, and the real bus state is only visible as `Link State` on the Ethernet side.

The `ComM` has support for this kind of connection between different `ComM Channels`, the pattern being mainly used for VLANs that shall not have separate network management. This support comes in the form of a `managing channel` (here the `ComM Channel` linked to the `CanController`) and one or more `managed channels` (here all Ethernet channels, including all VLANs). The `managed channels` refer to the `managing channel` via the `ComMManageReference`.

## 10.4 Published Information

For details refer to [4] Chapter 10.3 *"Published Information"*.

# A Not applicable requirements

**[CP_SWS_CanXL_00999]** ⌈These requirements are not applicable to this specification.⌋

# B Change History

Please note that the lists in this chapter also include constraints and specification items that have been removed from the specification in a later version. These constraints and specification items do not appear as hyperlinks in the document.

## B.1 Change History of this document according to AUTOSAR Release R22-11

### B.1.1 Added Specification Items in R22-11

| Number | Heading |
|---|---|
| [CP_SWS_CanXL_00001] | |
| [CP_SWS_CanXL_00002] | |
| [CP_SWS_CanXL_00003] | |
| [CP_SWS_CanXL_00004] | |
| [CP_SWS_CanXL_00005] | |
| [CP_SWS_CanXL_00006] | |
| [CP_SWS_CanXL_00007] | |
| [CP_SWS_CanXL_00008] | |
| [CP_SWS_CanXL_00009] | |
| [CP_SWS_CanXL_00010] | |
| [CP_SWS_CanXL_00011] | |
| [CP_SWS_CanXL_00012] | |
| [CP_SWS_CanXL_00013] | |
| [CP_SWS_CanXL_00020] | |
| [CP_SWS_CanXL_00021] | |
| [CP_SWS_CanXL_00022] | |
| [CP_SWS_CanXL_00023] | |
| [CP_SWS_CanXL_00024] | |
| [CP_SWS_CanXL_00025] | |
| [CP_SWS_CanXL_00026] | |
| [CP_SWS_CanXL_00027] | |
| [CP_SWS_CanXL_00028] | |
| [CP_SWS_CanXL_00029] | |
| [CP_SWS_CanXL_00031] | |
| [CP_SWS_CanXL_00032] | |
| [CP_SWS_CanXL_00034] | |
| [CP_SWS_CanXL_00036] | |

▽

△

| Number | Heading |
|---|---|
| [CP_SWS_CanXL_00037] | |
| [CP_SWS_CanXL_00039] | |
| [CP_SWS_CanXL_00040] | |
| [CP_SWS_CanXL_00042] | |
| [CP_SWS_CanXL_00043] | |
| [CP_SWS_CanXL_00044] | |
| [CP_SWS_CanXL_00046] | |
| [CP_SWS_CanXL_00047] | |
| [CP_SWS_CanXL_00048] | |
| [CP_SWS_CanXL_00058] | |
| [CP_SWS_CanXL_00059] | |
| [CP_SWS_CanXL_00060] | |
| [CP_SWS_CanXL_00061] | |
| [CP_SWS_CanXL_00063] | |
| [CP_SWS_CanXL_00064] | |
| [CP_SWS_CanXL_00065] | |
| [CP_SWS_CanXL_00066] | |
| [CP_SWS_CanXL_00067] | |
| [CP_SWS_CanXL_00068] | |
| [CP_SWS_CanXL_00069] | |
| [CP_SWS_CanXL_00073] | |
| [CP_SWS_CanXL_00075] | |
| [CP_SWS_CanXL_00076] | |
| [CP_SWS_CanXL_00077] | |
| [CP_SWS_CanXL_00078] | |
| [CP_SWS_CanXL_00081] | |
| [CP_SWS_CanXL_00082] | |
| [CP_SWS_CanXL_00083] | |
| [CP_SWS_CanXL_00084] | |
| [CP_SWS_CanXL_00085] | |
| [CP_SWS_CanXL_00086] | |
| [CP_SWS_CanXL_00087] | |
| [CP_SWS_CanXL_00089] | |
| [CP_SWS_CanXL_00090] | |
| [CP_SWS_CanXL_00091] | |
| [CP_SWS_CanXL_00092] | |
| [CP_SWS_CanXL_00096] | |
| [CP_SWS_CanXL_00097] | |

▽

△

| Number | Heading |
|---|---|
| [CP_SWS_CanXL_00098] | |
| [CP_SWS_CanXL_00103] | |
| [CP_SWS_CanXL_00104] | |
| [CP_SWS_CanXL_00106] | |
| [CP_SWS_CanXL_00107] | |
| [CP_SWS_CanXL_00108] | |
| [CP_SWS_CanXL_00109] | |
| [CP_SWS_CanXL_00110] | |
| [CP_SWS_CanXL_00112] | |
| [CP_SWS_CanXL_00113] | |
| [CP_SWS_CanXL_00114] | |
| [CP_SWS_CanXL_00116] | |
| [CP_SWS_CanXL_00117] | |
| [CP_SWS_CanXL_00118] | |
| [CP_SWS_CanXL_00119] | |
| [CP_SWS_CanXL_00120] | |
| [CP_SWS_CanXL_00121] | |
| [CP_SWS_CanXL_00122] | |
| [CP_SWS_CanXL_00123] | |
| [CP_SWS_CanXL_00124] | |
| [CP_SWS_CanXL_00125] | |
| [CP_SWS_CanXL_00126] | |
| [CP_SWS_CanXL_00127] | |
| [CP_SWS_CanXL_00128] | |
| [CP_SWS_CanXL_00129] | |
| [CP_SWS_CanXL_00130] | |
| [CP_SWS_CanXL_00131] | |
| [CP_SWS_CanXL_00132] | |
| [CP_SWS_CanXL_00999] | |
| [CP_SWS_CanXL_10001] | |
| [CP_SWS_CanXL_10002] | |
| [CP_SWS_CanXL_10003] | |
| [CP_SWS_CanXL_10004] | |
| [CP_SWS_CanXL_10005] | |
| [CP_SWS_CanXL_10006] | |
| [CP_SWS_CanXL_10007] | |
| [CP_SWS_CanXL_10008] | |
| [CP_SWS_CanXL_10009] | |

▽

△

| Number | Heading |
|---|---|
| [CP_SWS_CanXL_10010] | |
| [CP_SWS_CanXL_10011] | |
| [CP_SWS_CanXL_10012] | |
| [CP_SWS_CanXL_10013] | |
| [CP_SWS_CanXL_10014] | |
| [CP_SWS_CanXL_10015] | |
| [CP_SWS_CanXL_10016] | |
| [CP_SWS_CanXL_10017] | |
| [CP_SWS_CanXL_10018] | |
| [CP_SWS_CanXL_10020] | |
| [CP_SWS_CanXL_10021] | |
| [CP_SWS_CanXL_10022] | |
| [CP_SWS_CanXL_10023] | |
| [CP_SWS_CanXL_10024] | |
| [CP_SWS_CanXL_10025] | |
| [CP_SWS_CanXL_10026] | |
| [CP_SWS_CanXL_10027] | |
| [CP_SWS_CanXL_CONSTR_00001] | |

**Table B.1: Added Specification Items in R22-11**

### B.1.2 Changed Specification Items in R22-11

none

### B.1.3 Deleted Specification Items in R22-11

none

### B.1.4 Added Constraints in R22-11

none

### B.1.5 Changed Constraints in R22-11

none

### B.1.6 Deleted Constraints in R22-11

none

## B.2 Change History of this document according to AUTOSAR Release R23-11

### B.2.1 Added Specification Items in R23-11

| Number | Heading |
|---|---|
| [CP_SWS_CanXL_00133] | |
| [CP_SWS_CanXL_00134] | |
| [CP_SWS_CanXL_00135] | |
| [CP_SWS_CanXL_00136] | |
| [CP_SWS_CanXL_00137] | |
| [CP_SWS_CanXL_00138] | |
| [CP_SWS_CanXL_90000] | Definition of API function CanXL_GetCurrentTimeTuple |

**Table B.2: Added Specification Items in R23-11**

### B.2.2 Changed Specification Items in R23-11

| Number | Heading |
|---|---|
| [CP_SWS_CanXL_00025] | |
| [CP_SWS_CanXL_00026] | |
| [CP_SWS_CanXL_00027] | |
| [CP_SWS_CanXL_00028] | |
| [CP_SWS_CanXL_00029] | |
| [CP_SWS_CanXL_10006] | Definition of API function CanXL_GetCurrentTime |
| [CP_SWS_CanXL_10007] | Definition of API function CanXL_GetEgressTimeStamp |
| [CP_SWS_CanXL_10008] | Definition of API function CanXL_GetIngressTimeStamp |
| [CP_SWS_CanXL_10021] | Definition of optional interfaces in module CanXL |
| [CP_SWS_CanXL_10023] | Definition of imported datatypes of module CanXL |
| [CP_SWS_CanXL_10024] | Definiton of development errors in module CanXL |

**Table B.3: Changed Specification Items in R23-11**

### B.2.3 Deleted Specification Items in R23-11

none

### B.2.4 Added Constraints in R23-11

none

### B.2.5 Changed Constraints in R23-11

none

### B.2.6 Deleted Constraints in R23-11

none

## B.3 Change History of this document according to AUTOSAR Release R24-11

### B.3.1 Added Specification Items in R24-11

| Number | Heading |
|---|---|
| [CP_SWS_CanXL_00139] | Value range of the returned buffer index |
| [SWS_CanXL_91000] | Definition of API function CanXL_ReleaseRxBuffer |
| [SWS_CanXL_91002] | Definition of API function CanXL_ImmediateTransmit |

**Table B.4: Added Specification Items in R24-11**

### B.3.2 Changed Specification Items in R24-11

| Number | Heading |
|---|---|
| [CP_SWS_CanXL_10002] | Definition of API function CanXL_Write |
| [CP_SWS_CanXL_10003] | Definition of API function CanXL_Transmit |
| [CP_SWS_CanXL_10012] | Definition of API function CanXL_ProvideTxBuffer |
| [CP_SWS_CanXL_10013] | Definition of API function CanXL_UpdatePhysAddrFilter |
| [CP_SWS_CanXL_10014] | Definition of API function CanXL_TxConfirmation |
| [CP_SWS_CanXL_10016] | Definition of API function CanXL_SetControllerMode |
| [CP_SWS_CanXL_10017] | Definition of API function CanXL_GetControllerMode |
| [CP_SWS_CanXL_10018] | Definition of API function CanXL_GetPhysAddr |
| [CP_SWS_CanXL_10020] | Definition of API function CanXL_Receive |
| [CP_SWS_CanXL_10022] | Definition of mandatory interfaces required by module CanXL |
| [CP_SWS_CanXL_10023] | Definition of imported datatypes of module CanXL |
| [CP_SWS_CanXL_90000] | Definition of API function CanXL_GetCurrentTimeTuple |

**Table B.5: Changed Specification Items in R24-11**

### B.3.3 Deleted Specification Items in R24-11

none

### B.3.4 Added Constraints in R24-11

none

### B.3.5 Changed Constraints in R24-11

none

### B.3.6 Deleted Constraints in R24-11

none

## B.4 Change History of this document according to AUTOSAR Release R25-11

### B.4.1 Added Specification Items in R25-11

none

### B.4.2 Changed Specification Items in R25-11

| Number | Heading |
|---|---|
| [CP_SWS_CanXL_00003] | |
| [CP_SWS_CanXL_00092] | |
| [CP_SWS_CanXL_00121] | |
| [CP_SWS_CanXL_10003] | Definition of API function CanXL_Transmit |
| [CP_SWS_CanXL_10013] | Definition of API function CanXL_UpdatePhysAddrFilter |
| [CP_SWS_CanXL_10016] | Definition of API function CanXL_SetControllerMode |
| [CP_SWS_CanXL_10022] | Definition of mandatory interfaces required by module CanXL |
| [CP_SWS_CanXL_10023] | Definition of imported datatypes of module CanXL |
| [ECUC_Can_00499] | Definition of EcucParamConfContainerDef CanXLController |
| [ECUC_Can_00500] | Definition of EcucIntegerParamDef CanXLCtrlEthDefaultPriority |
| [ECUC_Can_00501] | Definition of EcucIntegerParamDef CanXLEthDefaultQueue |
| [ECUC_Can_00502] | Definition of EcucParamConfContainerDef CanXLEthEgress Fifo |
| [ECUC_Can_00503] | Definition of EcucIntegerParamDef CanXLEthEgressFifoCan XLPriority |
| [ECUC_Can_00504] | Definition of EcucIntegerParamDef CanXLEthEgressFifoCan XLQueue |
| [ECUC_Can_00505] | Definition of EcucIntegerParamDef CanXLEthEgressFifoIdx |
| [ECUC_Can_00506] | Definition of EcucStringParamDef CanXLEthPhysAddress |
| [ECUC_Can_00507] | Definition of EcucParamConfContainerDef CanXLEthIngress Fifo |
| [ECUC_Can_00508] | Definition of EcucIntegerParamDef CanXLEthIngressFifoIdx |
| [ECUC_Can_00509] | Definition of EcucIntegerParamDef CanXLEthIngressFifoCan XLQueue |
| [ECUC_Can_00510] | Definition of EcucIntegerParamDef CanXLEthIngressFifoVcid |
| [ECUC_Can_00511] | Definition of EcucReferenceDef CanXLEthEcucPartitionRef |
| [ECUC_Can_00512] | Definition of EcucParamConfContainerDef CanXLBaudrate Config |
| [ECUC_Can_00513] | Definition of EcucFloatParamDef CanXLBaudRate |
| [ECUC_Can_00514] | Definition of EcucIntegerParamDef CanXLPwmL |
| [ECUC_Can_00515] | Definition of EcucIntegerParamDef CanXLPwmS |



△

| Number | Heading |
|---|---|
| [ECUC_Can_00516] | Definition of EcucIntegerParamDef CanXLPwmO |
| [ECUC_Can_00517] | Definition of EcucIntegerParamDef CanXLPropSeg |
| [ECUC_Can_00518] | Definition of EcucIntegerParamDef CanXLSeg1 |
| [ECUC_Can_00519] | Definition of EcucIntegerParamDef CanXLSeg2 |
| [ECUC_Can_00520] | Definition of EcucIntegerParamDef CanXLSyncJumpWidth |
| [ECUC_Can_00521] | Definition of EcucIntegerParamDef CanXLSspOffset |
| [ECUC_Can_00522] | Definition of EcucBooleanParamDef CanXLTrcvPwmMode |
| [ECUC_Can_00523] | Definition of EcucBooleanParamDef CanXLErrorSignaling |
| [ECUC_Can_00524] | Definition of EcucParamConfContainerDef CanXLGeneral |
| [ECUC_Can_00525] | Definition of EcucBooleanParamDef CanXLEthGlobalTime Support |
| [ECUC_Can_00526] | Definition of EcucParamConfContainerDef CanXLHardware Object |
| [ECUC_Can_00527] | Definition of EcucIntegerParamDef CanXLObjectId |
| [ECUC_Can_00528] | Definition of EcucParamConfContainerDef CanXLHwFilter |
| [SWS_CanXL_91000] | Definition of API function CanXL_ReleaseRxBuffer |
| [SWS_CanXL_91002] | Definition of API function CanXL_ImmediateTransmit |

**Table B.6: Changed Specification Items in R25-11**

### B.4.3 Deleted Specification Items in R25-11

| Number | Heading |
|---|---|
| [CP_SWS_CanXL_00025] | |
| [CP_SWS_CanXL_00026] | |
| [CP_SWS_CanXL_00027] | |
| [CP_SWS_CanXL_00028] | |
| [CP_SWS_CanXL_00029] | |
| [CP_SWS_CanXL_10006] | Definition of API function CanXL_GetCurrentTime |

**Table B.7: Deleted Specification Items in R25-11**

### B.4.4 Added Constraints in R25-11

none

### B.4.5 Changed Constraints in R25-11

none

### B.4.6 Deleted Constraints in R25-11

none

## B.5 Change History of this document according to AUTOSAR Release R26-11

### B.5.1 Added Specification Items in R26-11

| Number | Heading |
|---|---|
| [CP_SWS_CanXL_00140] | |
| [CP_SWS_CanXL_00141] | |
| [CP_SWS_CanXL_00142] | |
| [ECUC_Can_00539] | Definition of EcucIntegerParamDef CanXLCtrlRequired Alignment |
| [ECUC_Can_00540] | Definition of EcucParamConfContainerDef CanXLCanSec Config |
| [ECUC_Can_00541] | Definition of EcucParamConfContainerDef CanXLCanSec Bypasses |
| [ECUC_Can_00542] | Definition of EcucParamConfContainerDef CanXLCanSec SecurityEntity |
| [ECUC_Can_00543] | Definition of EcucIntegerParamDef CanXLCanSecSecurity EntityIndex |
| [ECUC_Can_00544] | Definition of EcucParamConfContainerDef CanXLCanSec SecureChannel |
| [ECUC_Can_00545] | Definition of EcucIntegerParamDef CanXLCanSecSecure ChannelCSCI |
| [ECUC_Can_00546] | Definition of EcucParamConfContainerDef CanXLCanSecTxSc Filter |
| [ECUC_Can_00547] | Definition of EcucParamConfContainerDef CanXLCanSecFilter Range |
| [ECUC_Can_00548] | Definition of EcucParamConfContainerDef CanXLCanSecFilter RangeAcceptanceField |
| [ECUC_Can_00549] | Definition of EcucParamConfContainerDef CanXLCanSecFilter RangePriorityId |
| [ECUC_Can_00550] | Definition of EcucIntegerParamDef CanXLCanSecFilterRange AcceptanceFieldBase |
| [ECUC_Can_00551] | Definition of EcucIntegerParamDef CanXLCanSecFilterRange AcceptanceFieldMask |
| [ECUC_Can_00552] | Definition of EcucIntegerParamDef CanXLCanSecFilterRange PriorityIdBase |
| [ECUC_Can_00553] | Definition of EcucIntegerParamDef CanXLCanSecFilterRange PriorityIdMask |
| [ECUC_Can_00554] | Definition of EcucParamConfContainerDef CanXLCanSecFilter RangeVcid |
| [ECUC_Can_00555] | Definition of EcucParamConfContainerDef CanXLCanSecFilter RangeSduType |
| [ECUC_Can_00556] | Definition of EcucIntegerParamDef CanXLCanSecFilterRange SduTypeBase |



△

| Number | Heading |
|---|---|
| [ECUC_Can_00557] | Definition of EcucIntegerParamDef CanXLCanSecFilterRange SduTypeMask |
| [ECUC_Can_00558] | Definition of EcucIntegerParamDef CanXLCanSecFilterRange VcidBase |
| [ECUC_Can_00559] | Definition of EcucIntegerParamDef CanXLCanSecFilterRange VcidMask |
| [SWS_CanXL_91003] | Definition of API function CanXL_XLReleaseRxBuffer |

**Table B.8: Added Specification Items in R26-11**

### B.5.2 Changed Specification Items in R26-11

| Number | Heading |
|---|---|
| [CP_SWS_CanXL_00001] | |
| [CP_SWS_CanXL_00002] | |
| [CP_SWS_CanXL_00003] | |
| [CP_SWS_CanXL_00004] | |
| [CP_SWS_CanXL_00005] | |
| [CP_SWS_CanXL_00006] | |
| [CP_SWS_CanXL_00007] | |
| [CP_SWS_CanXL_00008] | |
| [CP_SWS_CanXL_00009] | |
| [CP_SWS_CanXL_00010] | |
| [CP_SWS_CanXL_00011] | |
| [CP_SWS_CanXL_00012] | |
| [CP_SWS_CanXL_00013] | |
| [CP_SWS_CanXL_00059] | |
| [CP_SWS_CanXL_00092] | |
| [CP_SWS_CanXL_00113] | |
| [CP_SWS_CanXL_00114] | |
| [CP_SWS_CanXL_00117] | |
| [CP_SWS_CanXL_00118] | |
| [CP_SWS_CanXL_00121] | |
| [CP_SWS_CanXL_00122] | |
| [CP_SWS_CanXL_00128] | |
| [CP_SWS_CanXL_10002] | Definition of API function CanXL_Write |
| [CP_SWS_CanXL_10003] | Definition of API function CanXL_Transmit |
| [CP_SWS_CanXL_10008] | Definition of API function CanXL_GetIngressTimeStamp |

▽

△

| Number | Heading |
|---|---|
| [CP_SWS_CanXL_10012] | Definition of API function CanXL_ProvideTxBuffer |
| [CP_SWS_CanXL_10013] | Definition of API function CanXL_UpdatePhysAddrFilter |
| [CP_SWS_CanXL_10023] | Definition of imported datatypes of module CanXL |
| [ECUC_Can_00499] | Definition of EcucParamConfContainerDef CanXLController |
| [ECUC_Can_00502] | Definition of EcucParamConfContainerDef CanXLEthEgress Fifo |
| [ECUC_Can_00503] | Definition of EcucIntegerParamDef CanXLEthEgressFifoCan XLPriority |
| [ECUC_Can_00504] | Definition of EcucIntegerParamDef CanXLEthEgressFifoCan XLQueue |
| [ECUC_Can_00505] | Definition of EcucIntegerParamDef CanXLEthEgressFifoIdx |
| [ECUC_Can_00506] | Definition of EcucStringParamDef CanXLEthPhysAddress |
| [ECUC_Can_00507] | Definition of EcucParamConfContainerDef CanXLEthIngress Fifo |
| [ECUC_Can_00508] | Definition of EcucIntegerParamDef CanXLEthIngressFifoIdx |
| [ECUC_Can_00509] | Definition of EcucIntegerParamDef CanXLEthIngressFifoCan XLQueue |
| [ECUC_Can_00510] | Definition of EcucIntegerParamDef CanXLEthIngressFifoVcid |
| [ECUC_Can_00511] | Definition of EcucReferenceDef CanXLEthEcucPartitionRef |
| [ECUC_Can_00512] | Definition of EcucParamConfContainerDef CanXLBaudrate Config |
| [ECUC_Can_00513] | Definition of EcucFloatParamDef CanXLBaudRate |
| [ECUC_Can_00514] | Definition of EcucIntegerParamDef CanXLPwmL |
| [ECUC_Can_00515] | Definition of EcucIntegerParamDef CanXLPwmS |
| [ECUC_Can_00516] | Definition of EcucIntegerParamDef CanXLPwmO |
| [ECUC_Can_00517] | Definition of EcucIntegerParamDef CanXLPropSeg |
| [ECUC_Can_00518] | Definition of EcucIntegerParamDef CanXLSeg1 |
| [ECUC_Can_00519] | Definition of EcucIntegerParamDef CanXLSeg2 |
| [ECUC_Can_00520] | Definition of EcucIntegerParamDef CanXLSyncJumpWidth |
| [ECUC_Can_00521] | Definition of EcucIntegerParamDef CanXLSspOffset |
| [ECUC_Can_00522] | Definition of EcucBooleanParamDef CanXLTrcvPwmMode |
| [ECUC_Can_00523] | Definition of EcucBooleanParamDef CanXLErrorSignaling |
| [ECUC_Can_00524] | Definition of EcucParamConfContainerDef CanXLGeneral |
| [ECUC_Can_00525] | Definition of EcucBooleanParamDef CanXLEthGlobalTime Support |
| [ECUC_Can_00526] | Definition of EcucParamConfContainerDef CanXLHardware Object |
| [ECUC_Can_00527] | Definition of EcucIntegerParamDef CanXLObjectId |
| [ECUC_Can_00528] | Definition of EcucParamConfContainerDef CanXLHwFilter |

▽

△

| Number | Heading |
|---|---|
| [SWS_CanXL_91002] | Definition of API function CanXL_ImmediateTransmit |

**Table B.9: Changed Specification Items in R26-11**

### B.5.3 Deleted Specification Items in R26-11

| Number | Heading |
|---|---|
| [CP_SWS_CanXL_00068] | |
| [CP_SWS_CanXL_00069] | |
| [CP_SWS_CanXL_00086] | |
| [CP_SWS_CanXL_00087] | |
| [CP_SWS_CanXL_00089] | |
| [CP_SWS_CanXL_00090] | |
| [CP_SWS_CanXL_00091] | |
| [CP_SWS_CanXL_00132] | |
| [CP_SWS_CanXL_00138] | |
| [CP_SWS_CanXL_10014] | Definition of API function CanXL_TxConfirmation |
| [CP_SWS_CanXL_10020] | Definition of API function CanXL_Receive |
| [ECUC_Can_00500] | Definition of EcucIntegerParamDef CanXLCtrlEthDefaultPriority |
| [ECUC_Can_00501] | Definition of EcucIntegerParamDef CanXLEthDefaultQueue |

**Table B.10: Deleted Specification Items in R26-11**

### B.5.4 Added Constraints in R26-11

none

### B.5.5 Changed Constraints in R26-11

none

### B.5.6 Deleted Constraints in R26-11

none