Document Title	Specification of Module XCP
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	412
Document Classification	Standard
Document Version	2.3.0
Document Status	Final
Part of Release	4.1
Revision	3

Document Change History

Date	Version	Changed by	Change Description
31.03.2014	2.3.0	AUTOSAR Release Management	• Editorial correction for faulty references links. • Minor editorial correction for SWS_Xcp_00841, SWS_Xcp_00844. • Changed Xcp_RxIndication argument from PduInfoType* to const PduInfoType*.
31.10.2013	2.2.0	AUTOSAR Release Management	• Minor corrections • Editorial changes • Removed chapter(s) on change documentation
26.02.2013	2.1.0	AUTOSAR Administration	• Reclassify XCP_E_INIT_FAILED from class production error to development
12.12.2011	2.0.0	AUTOSAR Administration	• Added parameters for Event Channel and Timestamp configuration • Added possibility to calculate memory consumption for ODT (DAQ & STIM) • Restructuring configuration parameters for static & dynamic ODT • Added support for deactivation of transmission capabilities
11.10.2010	1.1.0	AUTOSAR Administration	• Add chapter 7.8 (Version check), RTE limitation, OS Counter Ref • Remove InstanceID and known limitation (OS)
07.12.2009	1.0.0	AUTOSAR Administration	• Initial Release

Disclaimer

Advice for users

AUTOSAR specifications may contain exemplary items (exemplary reference models, "use cases", and/or references to exemplary technical solutions, devices, processes or software).

Any such exemplary items are contained in the specifications for illustration purposes only, and they themselves are not part of the AUTOSAR Standard. Neither their presence in such specifications, nor any later documentation of AUTOSAR conformance of products actually implementing such exemplary items, imply that intellectual property rights covering such exemplary items are licensed under the same rules as applicable to the AUTOSAR Standard.

Table of Contents

1 Introduction and functional overview

This specification specifies the functionality, API and the configuration of the AUTOSAR Basic Software module XCP

XCP is a protocol description (ASAM standard) between a master (tool) and a slave (device), which provides the following basic features:

- Synchronous data acquisition (measurement)
- Synchronous data stimulation (for rapid prototyping)
- Online memory calibration (read / write access)
- Calibration data page initialization and switching
- Flash Programming for ECU development purposes
- Every feature is optional and the access can be restricted
- Various communications busses are supported

XCP was designed according to the following principles:

- Minimal Slave resource consumption (RAM, ROM, runtime)
- Efficient communication
- Simple Slave implementation

2 Acronyms and abbreviations

Acronym:	Description:
AUTOSAR	**AUT**omotive **O**pen **S**ystem **AR**chitecture
A2L	File Extension for an ASAM 2MC Language File
ASAM	**A**ssociation for **S**tandardization of **A**utomation and **M**easuring Systems
BSW	**B**asic **Sof**t**w**are
CAN	**C**ontroller **A**rea **N**etwork
CanIf	**CAN** Inter**f**ace
CTO	**C**ommand **T**ransfer **O**bject
DAQ	**D**ata **A**c**Q**uisition, Data AcQuisition Packet
DTO	**D**ata **T**ransfer **O**bject
ECU	**E**lectronic **C**ontrol **U**nit
FrIf	**F**lex**R**ay **I**nter**f**ace
HIS	**H**ersteller **I**nitiative **S**oftware
LPDU	Data Link Layer PDU
MCD	**M**easurement **C**alibration and **D**iagnostics
MISRA	**M**otor **I**ndustry **S**oftware **R**eliability **A**ssociation
ODT	**O**bject **D**escriptor **T**able
PDU	**P**rotocol **D**ata **U**nit
RAM	**R**andom **A**ccess **M**emory
ROM	**R**ead **O**nly **M**emory
SchM	**Sch**edule **M**anager
SVN	**S**ub**v**ersio**n**
SRS	**S**oftware **R**equirements **S**pecification
STIM	Data **Stim**ulation packet
SW	**S**oft**w**are
SWS	**S**oft**w**are **S**pecification
TCP/IP	**T**ransfer **C**ontrol **P**rotocol **/ I**nternet **P**rotocol
TS	**T**ime **S**tamp
UDP/IP	**U**ser **D**atagram **P**rotocol **/ I**nternet **P**rotocol
URL	**U**niform **R**esource **L**ocator
XCP	Universal Calibration Protocol
XML	E**x**tensible **M**arkup **L**anguage
ISR	**I**nterrupt **S**ervice **R**outine
DEM	**D**iagnostic **E**vent **M**anager (AUTOSAR BSW module)
DET	**D**evelopment **E**rror **T**racer (AUTOSAR BSW module)

3 Related documentation

3.1 Input documents

[0] Basic Software Module Description Template
AUTOSAR_TPS_BSWModuleDescriptionTemplate.pdf

[1] List of Basic Software Modules
AUTOSAR_TR_BSWModuleList.pdf

[2] AUTOSAR Layered Software Architecture
AUTOSAR_EXP_LayeredSoftwareArchitecture.pdf

[3] General Requirements on Basic Software Modules
AUTOSAR_SRS_BSWGeneral.pdf

[4] Specification of RTE (BSW Scheduler)
AUTOSAR_SWS_RTE.pdf

[5] Specification of ECU Configuration
AUTOSAR_TPS_ECUConfiguration

[6] Specification of Memory Mapping
AUTOSAR_SWS_MemoryMapping.pdf

[7] Specification of FlexRay Interface
AUTOSAR_SWS_FlexRayInterface.pdf

[8] Specification of CAN Interface
AUTOSAR_SWS_CANInterface

[9] Specification of Socket Adaptor
AUTOSAR_SWS_SocketAdaptor

[10] Requirements on XCP Module
AUTOSAR_SRS_XCP.pdf

[11] AUTOSAR OS Specification
AUTOSAR_SWS_OS

[12] General Specification of Basic Software Modules
AUTOSAR_SWS_BSWGeneral.pdf

3.1.1 Related standards and norms

[13] ASAM XCP – The Universal Measurement and Calibration Protocol:
ASAM_XCP_Part1-Overview - Version 1.1

[14] ASAM XCP – Transport Layer Specification XCP on CAN:
ASAM_XCP_Part3 Transport-Layer-Specification_XCPonCAN - Version 1.1

[15] ASAM XCP – Transport Layer Specification XCP on Ethernet: ASAM_XCP_Part3-Transport-Layer-Specification_XCPonEthernet (TCP_IP&UDP_IP) – Version 1.1
[16] ASAM XCP – Transport Layer Specification XCP on FlexRay: ASAM_XCP_Part3-Transport-Layer-Specification_XCPonFlexRay-Version 1.1

3.2 Related specification

AUTOSAR provides a General Specification on Basic Software modules [12] (SWS BSW General), which is also valid for XCP.

Thus, the specification SWS BSW General shall be considered as additional and required specification for XCP.

4 Constraints and assumptions

4.1 Limitations

The following XCP features are currently out of scope:

- The XCP feature “Flash Programming for ECU development purposes” is currently out of AUTOSAR scope!

- The SET_DAQ_ID command according to the XCP CAN Transport Layer Specification is not part of the AUTOSAE XCP module”

- Currently, the AUTOSAR RTE does not offer APIs for direct communication with XCP

- For further details concerning the supported feature set, please refer to [13]

- NAX is only configurable through the ASAM configuration file A2L.

Please note:
For the communications bus LIN, no ASAM XCP is specified.

4.2 Applicability to car domains

n/a

5 Dependencies to other modules

This section describes the relations to other modules and files within the AUTOSAR basic software architecture. It contains brief descriptions of configuration information and services, which are required by the XCP module from other modules.

5.1 AUTOSAR RTE (BSW Scheduler)

The BSW Scheduler calls the main functions of the Xcp, which are necessary for the cyclic processes of the Xcp.

5.2 AUTOSAR FlexRay Interface

The FlexRay Interface is used to transmit and receive XCP PDUs via FlexRay.

5.3 AUTOSAR CAN Interface

The CAN Interface is used to transmit and receive XCP PDUs via CAN.

5.4 AUTOSAR SocketAdaptor

The SocketAdaptor is used to transmit and receive XCP PDUs via Ethernet.

5.5 AUTOSAR RTE

The RTE is used for copying calibration parameters from ROM/FLASH to RAM and to use the double pointered method

5.6 AUTOSAR OS

In order to be able to use the time stamped feature of XCP, an AUTOSAR OS Counter is used.

5.7 AUTOSAR Diagnostic Event Manager

In order to be able to report production errors, the XCP has to have access to the Diagnostic Event Manager.

5.8 AUTOSAR Development Error Tracer

In order to be able to report development errors, the XCP has to have access to the error hook of the Development Error Tracer.

5.9 File structure

5.9.1 Code file structure

[**SWS_Xcp_00501**]

⌈The code file structure shall not be defined within this specification completely. At this point it shall be pointed out that the code-file structure shall include the following files named:

- Xcp.c – general source code file of the module XCP
- Xcp_Cfg.c – for pre-compile time configurable parameters
- Xcp_Lcfg.c – for link time configurable parameters and
- Xcp_PBcfg.c – for post build time configurable parameters. ⌋ (SRS_BSW_00380, SRS_BSW_00419, SRS_BSW_00383, SRS_BSW_00346, SRS_BSW_00158)

These files shall contain all link time and post-build time configurable parameters.

[**SWS_Xcp_00500**]

⌈The module XCP shall access the location of the API of all used modules for pre-compile time configuration by either using of external declaration in includes of the used modules' public header files <x>.h or by the code file `Xcp_Cfg.c`.⌋()

5.9.2 Header file structure

[**SWS_Xcp_00502**] ⌈

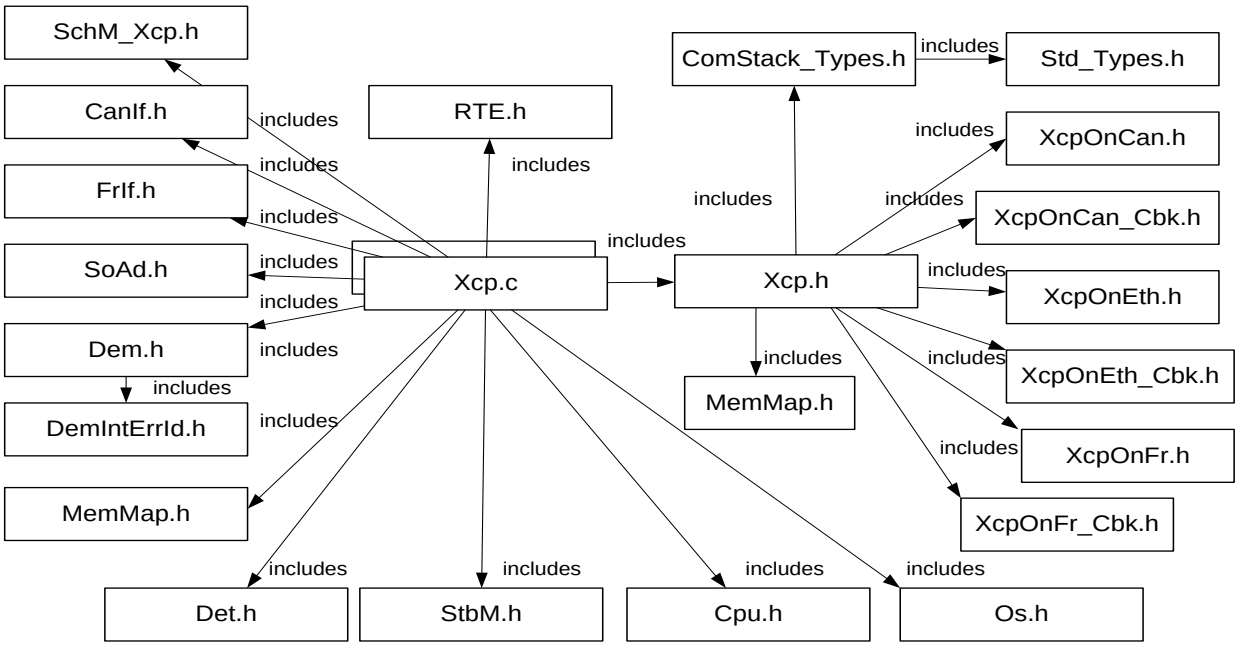

Figure 1: XCP Header File Structure

」(SRS_BSW_00381, SRS_BSW_00412, SRS_BSW_00409, SRS_BSW_00301)

The XCP module shall include the *Dem.h* file. By this inclusion, the APIs to report errors as well as the required Event Id symbols are included. This specification defines the name of the Event Id symbols, which are provided by XML to the DEM configuration tool. The DEM configuration tool assigns ECU dependent values to the Event Id symbols and publishes the symbols *in Dem_IntErrId.h*.

[**SWS_Xcp_00505**]
「The implementation of the XCP module shall provide the header file Xcp.*h*, which is the main module interface file. It shall contain all types and function prototypes required by the XCP module's environment.」(SRS_BSW_00302)

[**SWS_Xcp_00506**]
「The implementation of the XCP on CAN module shall provide the header file `XcpOnCan_Cfg.h` that shall contain the pre-compile-time configuration parameters.」
()

[**SWS_Xcp_00507**] 「
The implementation of the XCP on FlexRay module shall provide the header file `XcpOnFr_Cfg.h` that shall contain the pre-compile-time configuration parameters.」()

[**SWS_Xcp_00508**] 「
The implementation of the XCP on Ethernet module shall provide the header file `XcpOnEth_Cfg.h` that shall contain the pre-compile-time configuration parameters.」
()

6 Requirements traceability

Requirement	Description	Satisfied by
-	-	SWS_Xcp_00500
-	-	SWS_Xcp_00506
-	-	SWS_Xcp_00507
-	-	SWS_Xcp_00508
-	-	SWS_Xcp_00708
-	-	SWS_Xcp_00715
-	-	SWS_Xcp_00716
-	-	SWS_Xcp_00718
-	-	SWS_Xcp_00721
-	-	SWS_Xcp_00722
-	-	SWS_Xcp_00723
-	-	SWS_Xcp_00724
-	-	SWS_Xcp_00725
-	-	SWS_Xcp_00726
-	-	SWS_Xcp_00728
-	-	SWS_Xcp_00729
-	-	SWS_Xcp_00730
-	-	SWS_Xcp_00731
-	-	SWS_Xcp_00732
-	-	SWS_Xcp_00735
-	-	SWS_Xcp_00736
-	-	SWS_Xcp_00737
-	-	SWS_Xcp_00738
-	-	SWS_Xcp_00739
-	-	SWS_Xcp_00740
-	-	SWS_Xcp_00764
-	-	SWS_Xcp_00768
-	-	SWS_Xcp_00801
-	-	SWS_Xcp_00802
-	-	SWS_Xcp_00813
-	-	SWS_Xcp_00814
-	-	SWS_Xcp_00824
-	-	SWS_Xcp_00825
-	-	SWS_Xcp_00831
-	-	SWS_Xcp_00832
-	-	SWS_Xcp_00833

-	-	SWS_Xcp_00834
-	-	SWS_Xcp_00835
-	-	SWS_Xcp_00836
-	-	SWS_Xcp_00840
-	-	SWS_Xcp_00841
-	-	SWS_Xcp_00842
-	-	SWS_Xcp_00843
-	-	SWS_Xcp_00844
-	-	SWS_Xcp_00845
-	-	SWS_Xcp_00846
-	-	SWS_Xcp_00847
-	-	SWS_Xcp_00848
-	-	SWS_Xcp_00849
-	-	SWS_Xcp_00850
BSW00431	-	SWS_Xcp_00999
BSW00434	-	SWS_Xcp_00999
BSW42911	-	SWS_Xcp_00711
SRS_BSW_00003	All software modules shall provide version and identification information	SWS_Xcp_00807
SRS_BSW_00005	Modules of the æC Abstraction Layer (MCAL) may not have hard coded horizontal interfaces	SWS_Xcp_00999
SRS_BSW_00006	The source code of software modules above the æC Abstraction Layer (MCAL) shall not be processor and compiler dependent.	SWS_Xcp_00999
SRS_BSW_00009	All Basic SW Modules shall be documented according to a common standard.	SWS_Xcp_00999
SRS_BSW_00010	The memory consumption of all Basic SW Modules shall be documented for a defined configuration for all supported platforms.	SWS_Xcp_00999
SRS_BSW_00101	The Basic Software Module shall be able to initialize variables and hardware in a separate initialization function	SWS_Xcp_00803
SRS_BSW_00158	All modules of the AUTOSAR Basic Software shall strictly separate configuration from implementation	SWS_Xcp_00501
SRS_BSW_00159	All modules of the AUTOSAR Basic Software shall support a tool based configuration	SWS_Xcp_00102
SRS_BSW_00161	The AUTOSAR Basic Software shall provide a microcontroller abstraction layer which provides a standardized interface to higher software layers	SWS_Xcp_00999
SRS_BSW_00162	The AUTOSAR Basic Software shall provide a hardware abstraction layer	SWS_Xcp_00999
SRS_BSW_00164	The Implementation of interrupt service routines shall be done by the Operating System, complex drivers or modules	SWS_Xcp_00999

SRS_BSW_00167	All AUTOSAR Basic Software Modules shall provide configuration rules and constraints to enable plausibility checks	SWS_XCP_00103, SWS_Xcp_00104, SWS_Xcp_00105
SRS_BSW_00168	SW components shall be tested by a function defined in a common API in the Basis-SW	SWS_Xcp_00999
SRS_BSW_00170	The AUTOSAR SW Components shall provide information about their dependency from faults, signal qualities, driver demands	SWS_Xcp_00999
SRS_BSW_00171	Optional functionality of a Basic-SW component that is not required in the ECU shall be configurable at pre-compile-time	SWS_Xcp_00999
SRS_BSW_00172	The scheduling strategy that is built inside the Basic Software Modules shall be compatible with the strategy used in the system	SWS_Xcp_00999
SRS_BSW_00301	All AUTOSAR Basic Software Modules shall only import the necessary information	SWS_Xcp_00502
SRS_BSW_00302	All AUTOSAR Basic Software Modules shall only export information needed by other modules	SWS_Xcp_00505
SRS_BSW_00306	AUTOSAR Basic Software Modules shall be compiler and platform independent	SWS_Xcp_00999
SRS_BSW_00309	All AUTOSAR Basic Software Modules shall indicate all global data with read-only purposes by explicitly assigning the const keyword	SWS_Xcp_00999
SRS_BSW_00312	Shared code shall be reentrant	SWS_Xcp_00999
SRS_BSW_00314	All internal driver modules shall separate the interrupt frame definition from the service routine	SWS_Xcp_00999
SRS_BSW_00318	Each AUTOSAR Basic Software Module file shall provide version numbers in the header file	SWS_Xcp_00807
SRS_BSW_00321	The version numbers of AUTOSAR Basic Software Modules shall be enumerated according specific rules	SWS_Xcp_00999
SRS_BSW_00325	The runtime of interrupt service routines and functions that are running in interrupt context shall be kept short	SWS_Xcp_00999
SRS_BSW_00326	-	SWS_Xcp_00999
SRS_BSW_00327	Error values naming convention	SWS_Xcp_00763
SRS_BSW_00328	All AUTOSAR Basic Software Modules shall avoid the duplication of code	SWS_Xcp_00999
SRS_BSW_00329	-	SWS_Xcp_00999
SRS_BSW_00330	It shall be allowed to use macros instead of functions where source code is used and runtime is critical	SWS_Xcp_00999
SRS_BSW_00331	All Basic Software Modules shall strictly separate error and status information	SWS_Xcp_00999
SRS_BSW_00333	For each callback function it shall be specified if it is called from interrupt context or not	SWS_Xcp_00999
SRS_BSW_00335	Status values naming convention	SWS_Xcp_00999

SRS_BSW_00336	Basic SW module shall be able to shutdown	SWS_Xcp_00999
SRS_BSW_00341	Module documentation shall contains all needed informations	SWS_Xcp_00999
SRS_BSW_00344	BSW Modules shall support link-time configuration	SWS_Xcp_00741
SRS_BSW_00345	BSW Modules shall support pre-compile configuration	SWS_Xcp_00742
SRS_BSW_00346	All AUTOSAR Basic Software Modules shall provide at least a basic set of module files	SWS_Xcp_00501
SRS_BSW_00347	A Naming seperation of different instances of BSW drivers shall be in place	SWS_Xcp_00999
SRS_BSW_00358	The return type of init() functions implemented by AUTOSAR Basic Software Modules shall be void	SWS_Xcp_00803
SRS_BSW_00360	AUTOSAR Basic Software Modules callback functions are allowed to have parameters	SWS_Xcp_00999
SRS_BSW_00370	-	SWS_Xcp_00999
SRS_BSW_00371	The passing of function pointers as API parameter is forbidden for all AUTOSAR Basic Software Modules	SWS_Xcp_00999
SRS_BSW_00373	The main processing function of each AUTOSAR Basic Software Module shall be named according the defined convention	SWS_Xcp_00823
SRS_BSW_00374	All Basic Software Modules shall provide a readable module vendor identification	SWS_Xcp_00807
SRS_BSW_00375	Basic Software Modules shall report wake-up reasons	SWS_Xcp_00999
SRS_BSW_00377	A Basic Software Module can return a module specific types	SWS_Xcp_00999
SRS_BSW_00379	All software modules shall provide a module identifier in the header file and in the module XML description file.	SWS_Xcp_00807
SRS_BSW_00380	Configuration parameters being stored in memory shall be placed into separate c-files	SWS_Xcp_00501
SRS_BSW_00381	The pre-compile time parameters shall be placed into a separate configuration header file	SWS_Xcp_00502
SRS_BSW_00383	The Basic Software Module specifications shall specify which other configuration files from other modules they use at least in the description	SWS_Xcp_00501
SRS_BSW_00387	The Basic Software Module specifications shall specify how the callback function is to be implemented	SWS_Xcp_00999
SRS_BSW_00401	Documentation of multiple instances of configuration parameters shall be available	SWS_Xcp_00999
SRS_BSW_00402	Each module shall provide version information	SWS_Xcp_00807
SRS_BSW_00404	BSW Modules shall support post-build configuration	SWS_Xcp_00742
SRS_BSW_00405	BSW Modules shall support multiple	SWS_Xcp_00803

	configuration sets	
SRS_BSW_00407	Each BSW module shall provide a function to read out the version information of a dedicated module implementation	SWS_Xcp_00807
SRS_BSW_00409	All production code error ID symbols are defined by the Dem module and shall be retrieved by the other BSW modules from Dem configuration	SWS_Xcp_00502
SRS_BSW_00410	Compiler switches shall have defined values	SWS_Xcp_00999
SRS_BSW_00411	All AUTOSAR Basic Software Modules shall apply a naming rule for enabling/disabling the existence of the API	SWS_Xcp_00807
SRS_BSW_00412	References to c-configuration parameters shall be placed into a separate h-file	SWS_Xcp_00502
SRS_BSW_00413	An index-based accessing of the instances of BSW modules shall be done	SWS_Xcp_00999
SRS_BSW_00414	The init function may have parameters	SWS_Xcp_00803
SRS_BSW_00415	Interfaces which are provided exclusively for one module shall be separated into a dedicated header file	SWS_Xcp_00999
SRS_BSW_00416	The sequence of modules to be initialized shall be configurable	SWS_Xcp_00999
SRS_BSW_00417	Software which is not part of the SW-C shall report error events only after the DEM is fully operational.	SWS_Xcp_00999
SRS_BSW_00419	If a pre-compile time configuration parameter is implemented as "const" it should be placed into a separate c-file	SWS_Xcp_00501
SRS_BSW_00423	BSW modules with AUTOSAR interfaces shall be describable with the means of the SW-C Template	SWS_Xcp_00999
SRS_BSW_00424	BSW module main processing functions shall not be allowed to enter a wait state	SWS_Xcp_00823
SRS_BSW_00425	The BSW module description template shall provide means to model the defined trigger conditions of schedulable objects	SWS_Xcp_00999
SRS_BSW_00426	BSW Modules shall ensure data consistency of data which is shared between BSW modules	SWS_Xcp_00999
SRS_BSW_00427	ISR functions shall be defined and documented in the BSW module description template	SWS_Xcp_00999
SRS_BSW_00428	A BSW module shall state if its main processing function(s) has to be executed in a specific order or sequence	SWS_Xcp_00999
SRS_BSW_00432	Modules should have separate main processing functions for read/receive and write/transmit data path	SWS_Xcp_00999
SRS_BSW_00433	Main processing functions are only allowed to be called from task bodies provided by the BSW Scheduler	SWS_Xcp_00823
SRS_Xcp_29001	The AUTOSAR XCP module shall be located	SWS_Xcp_00701

	above the bus interfaces / Socket Adaptor	
SRS_Xcp_29002	The AUTOSAR XCP shall make use of the data transmit- and receive APIs of the Bus Interfaces	SWS_Xcp_00712, SWS_Xcp_00714, SWS_Xcp_00720, SWS_Xcp_00734
SRS_Xcp_29003	The AUTOSAR XCP messages shall be identified by unique PDU-IDs	SWS_Xcp_00702
SRS_Xcp_29004	The XCP Specification Version 1.1 shall be used	SWS_Xcp_00703
SRS_Xcp_29005	XCP on CAN shall be supported	SWS_Xcp_00713
SRS_Xcp_29006	XCP on FlexRay shall be supported	SWS_Xcp_00719
SRS_Xcp_29007	XCP on Ethernet shall be supported	SWS_Xcp_00733
SRS_Xcp_29008	The code generator of the XCP Module shall generate the A2L IF_DATA section	SWS_Xcp_00999
SRS_Xcp_29009	The slave shall transfer the contents of the elements defined in each ODT of the DAQ-list to the master	SWS_Xcp_00705
SRS_Xcp_29010	Synchronous Data Stimulation shall be the inverse mode of Synchronous Data Acquisition	SWS_Xcp_00707
SRS_Xcp_29012	The XCP master shall already send the next request before having received the response on the previous request	SWS_Xcp_00710
SRS_Xcp_29013	It shall be possible to configure the DAQ Lists dynamically	SWS_Xcp_00706
SRS_Xcp_29014	It shall be possible to transmit a timestamp within the XCP packet	SWS_Xcp_00709
SRS_Xcp_29015	It shall be possible to bypass data by making use of Synchronous Data Acquisition and Synchronous Data Stimulation simultaneously	SWS_Xcp_00761
SRS_Xcp_29016	The feature "Seed&Key" shall be used for protection handling purpose	SWS_Xcp_00766
SRS_Xcp_29017	The AUTOSAR XCP module shall implement an interface for initialization.	SWS_Xcp_00803

General Requirements on Basic Software Modules

Requirement		***Satisfied by***
SRS_BSW_00344	Reference to link-time configuration	SWS_Xcp_00741
SRS_BSW_00404	Reference to post build time configuration	SWS_Xcp_00742
SRS_BSW_00405	Reference to multiple configuration sets	SWS_Xcp_00803
SRS_BSW_00345	Pre-compile-time configuration	SWS_Xcp_00742
SRS_BSW_00159	Tool-based configuration	SWS_Xcp_00102
SRS_BSW_00167	Static configuration checking	SWS_Xcp_00103 SWS_Xcp_00104 SWS_Xcp_00105
SRS_BSW_00171	Configurability of optional functionality	n/a
SRS_BSW_00170	Data for reconfiguration of AUTOSAR SW-Components	n/a
SRS_BSW_00380	Separate C-Files for configuration parameters	SWS_Xcp_00501
SRS_BSW_00419	Separate C-Files for pre-compile time configuration parameters	SWS_Xcp_00501
SRS_BSW_00381	Separate configuration header file for pre-compile time	SWS_Xcp_005012

	parameters	
SRS_BSW_00412	Separate H-File for configuration parameters	SWS_Xcp_00502
SRS_BSW_00383	List dependencies of configuration files	SWS_Xcp_00501
SRS_BSW_00384	List dependencies to other modules	Chapter 5
SRS_BSW_00387	Specify the configuration class of callback function	n/a
SRS_BSW_00388	Introduce containers	SWS_Xcp_00101
SRS_BSW_00389	Containers shall have names	Chapter 10.2
SRS_BSW_00390	Parameter content shall be unique within the module	Chapter 10.2
SRS_BSW_00391	Parameter shall have unique names	Chapter 10.2
SRS_BSW_00392	Parameters shall have a type	Chapter 10.2
SRS_BSW_00393	Parameters shall have a range	Chapter 10.2
SRS_BSW_00394	Specify the scope of the parameters	Chapter 10.2
SRS_BSW_00395	List the required parameters (per parameter)	Chapter 10.2
SRS_BSW_00396	Configuration classes	Chapter 10.2
SRS_BSW_00397	Pre-compile-time parameters	Chapter 10.2
SRS_BSW_00398	Link-time parameters	Chapter 10.2
SRS_BSW_00399	Loadable Post-build time parameters	Chapter 10.2
SRS_BSW_00400	Selectable Post-build time parameters	Chapter 10.2
SRS_BSW_00402	Published information	SWS_Xcp_00807
SRS_BSW_00375	Notification of wake-up reason	n/a
SRS_BSW_00101	Initialization interface	SWS_Xcp_00803
SRS_BSW_00416	Sequence of Initialization	n/a
SRS_BSW_00406	Check module initialization	Xcp811
SRS_BSW_00168	Diagnostic Interface of SW components	n/a
SRS_BSW_00407	Function to read out published parameters	SWS_Xcp_00807
SRS_BSW_00423	Usage of SW-C template to describe BSW modules with AUTOSAR Interfaces	n/a
SRS_BSW_00424	BSW main processing function task allocation	SWS_Xcp_00823
SRS_BSW_00425	Trigger conditions for schedulable objects	n/a
SRS_BSW_00426	Exclusive areas in BSW modules	n/a
SRS_BSW_00427	ISR description for BSW modules	n/a
SRS_BSW_00428	Execution order dependencies of main processing functions	n/a
SRS_BSW_00429	Restricted BSW OS functionality access	Chapter 5.6
BSW00431	The BSW Scheduler module implements task bodies	n/a
SRS_BSW_00432	Modules should have separate main processing functions for read/receive and write/transmit data path	n/a
SRS_BSW_00433	Calling of main processing functions	SWS_Xcp_00823
BSW00434	The Schedule Module shall provide an API for exclusive areas	n/a
SRS_BSW_00336	Shutdown interface	n/a
SRS_BSW_00337	Classification of errors	Error classification
SRS_BSW_00338	Detection and Reporting of development errors	Chapter 7.6 Chapter 7.7
SRS_BSW_00369	Do not return development error codes via API	Chapter
SRS_BSW_00339	Reporting of production relevant error status	Chapter
SRS_BSW_00417	Reporting of Error Events by Non-Basic Software	n/a
SRS_BSW_00323	API parameter checking	Chapter 8.3
SRS_BSW_00004	Version check	SWS_Xcp_00749
SRS_BSW_00409	Header files for production code error IDs	SWS_Xcp_00502
SRS_BSW_00385	List possible error notifications	Chapter 7.5
SRS_BSW_00386	Configuration for detecting an error	SWS_Xcp_00754
SRS_BSW_00161	Microcontroller abstraction	n/a
SRS_BSW_00162	ECU layout abstraction	n/a
SRS_BSW_00005	No hard coded horizontal interfaces within MCAL	n/a
SRS_BSW_00415	User dependent include files	n/a
SRS_BSW_00164	Implementation of interrupt service routines	n/a
SRS_BSW_00325	Runtime of interrupt service routines	n/a

SRS_BSW_00326	Transition from ISRs to OS tasks	n/a
SRS_BSW_00342	Usage of source code and object code	Chapter 10.2.1
SRS_BSW_00343	Specification and configuration of time	Chapter 10.2
SRS_BSW_00160	Human-readable configuration data	SWS_Xcp_00744
SRS_BSW_00007	HIS MISRA C	SWS_Xcp_00745
SRS_BSW_00300	Module naming convention	SWS_Xcp_00503
SRS_BSW_00413	Accessing instances of BSW modules	n/a
SRS_BSW_00347	Naming separation of different instances of BSW drivers	n/a
SRS_BSW_00305	Self-defined data types naming convention	Chapter 8.2
SRS_BSW_00307	Global variables naming convention	SWS_Xcp_00800
SRS_BSW_00310	API naming convention	SWS_Xcp_00800
SRS_BSW_00373	Main processing function naming convention	SWS_Xcp_00823
SRS_BSW_00327	Error values naming convention	SWS_Xcp_00763
SRS_BSW_00335	Status values naming convention	n/a
SRS_BSW_00350	Development error detection keyword	Xcp753
SRS_BSW_00408	Configuration parameter naming convention	SWS_Xcp_00800
SRS_BSW_00410	Compiler switches shall have defined values	n/a
SRS_BSW_00411	Get version info keyword	SWS_Xcp_00807 SWS_Xcp_00808 SWS_Xcp_00809 SWS_Xcp_00810
SRS_BSW_00346	Basic set of module files	SWS_Xcp_00501
SRS_BSW_00158	Separation of configuration from implementation	SWS_Xcp_00501
SRS_BSW_00314	Separation of interrupt frames and service routines	n/a
SRS_BSW_00370	Separation of callback interface from API	n/a
SRS_BSW_00348	Standard type header	Chapter 5.9
SRS_BSW_00353	Platform specific type header	Chapter 5.9
SRS_BSW_00361	Compiler specific language extension header	Chapter 5.9
SRS_BSW_00301	Limit imported information	SWS_Xcp_00502
SRS_BSW_00302	Limit exported information	SWS_Xcp_00505
SRS_BSW_00328	Avoid duplication of code	n/a
SRS_BSW_00312	Shared code shall be reentrant	n/a
SRS_BSW_00006	Platform independency	n/a
SRS_BSW_00357	Standard API return type	Chapter 8.3
SRS_BSW_00377	Module specific API return types	n/a
SRS_BSW_00304	AUTOSAR integer data types	Chapter 8.3
SRS_BSW_00355	Do not redefine AUTOSAR integer data types	Chapter 8.3
SRS_BSW_00378	AUTOSAR boolean type	Chapter 8.3
SRS_BSW_00306	Avoid direct use of compiler and platform specific keywords	n/a
SRS_BSW_00308	Definition of global data	SWS_Xcp_00760
SRS_BSW_00309	Global data with read-only constraint	n/a
SRS_BSW_00371	Do not pass function pointers via API	n/a
SRS_BSW_00358	Return type of init() functions	SWS_Xcp_00803
SRS_BSW_00414	Parameter of init function	SWS_Xcp_00803
SRS_BSW_00376	Return type and parameters of main processing functions	Chapter 8.5
SRS_BSW_00359	Return type of callback functions	Chapter 8.4
SRS_BSW_00360	Parameters of callback functions	n/a
SRS_BSW_00329	Avoidance of generic interfaces	n/a
SRS_BSW_00330	Usage of macros / inline functions instead of functions	n/a
SRS_BSW_00331	Separation of error and status values	n/a
SRS_BSW_00009	Module User Documentation	n/a
SRS_BSW_00401	Documentation of multiple instances of configuration parameters	n/a
SRS_BSW_00172	Compatibility and documentation of scheduling strategy	n/a
SRS_BSW_00010	Memory resource documentation	n/a
SRS_BSW_00333	Documentation of callback function context	n/a
SRS_BSW_00374	Module vendor identification	SWS_Xcp_00807

SRS_BSW_00379	Module identification	SWS_Xcp_00807
SRS_BSW_00003	Version identification	SWS_Xcp_00807
SRS_BSW_00318	Format of module version numbers	SWS_Xcp_00807
SRS_BSW_00321	Enumeration of module version numbers	n/a
SRS_BSW_00341	Microcontroller compatibility documentation	n/a
SRS_BSW_00334	Provision of XML file	SWS_Xcp_00751
SRS_BSW_00435	Header File Structure for the Basic Software Scheduler	SWS_Xcp_00747
SRS_BSW_00436	Module Header File Structure for the Memory Mapping	SWS_Xcp_00748

6.1 Requirements on the XCP Basic Software Module

Requirement		***Satisfied by***
SRS_Xcp_29001	Location of XCP within the architecture	SWS_Xcp_00701
SRS_Xcp_29002	API usage	SWS_Xcp_00712, SWS_Xcp_00714, SWS_Xcp_00720, SWS_Xcp_00734
SRS_Xcp_29003	Unique PDU-ID	SWS_Xcp_00702
SRS_Xcp_29004	XCP Specification Version 1.1	SWS_Xcp_00703
SRS_Xcp_29005	XCP on CAN	SWS_Xcp_00713
SRS_Xcp_29006	XCP on FlexRay	SWS_Xcp_00719
SRS_Xcp_29007	XCP on Ethernet	SWS_Xcp_00733
SRS_Xcp_29008	A2L Support	n/a
SRS_Xcp_29009	Synchronous data acquisition	SWS_Xcp_00705
SRS_Xcp_29010	Synchronous data stimulation	SWS_Xcp_00707
SRS_Xcp_29011	Block communication mode	SWS_Xcp_00711
SRS_Xcp_29012	Interleaved communication mode	SWS_Xcp_00710
SRS_Xcp_29013	Dynamic data transfer configuration	SWS_Xcp_00706
SRS_Xcp_29014	Timestamped Data transfer	SWS_Xcp_00709
SRS_Xcp_29015	Bypassing	SWS_Xcp_00759
SRS_Xcp_29016	Seed & Key	SWS_Xcp_00760
SRS_Xcp_29017	XCP Initialization	SWS_Xcp_00803

7 Functional specification

The specification of the module XCP shall define all parameters and interfaces, which are required to use the ASAM XCP protocol specification within an AUTOSAR environment.

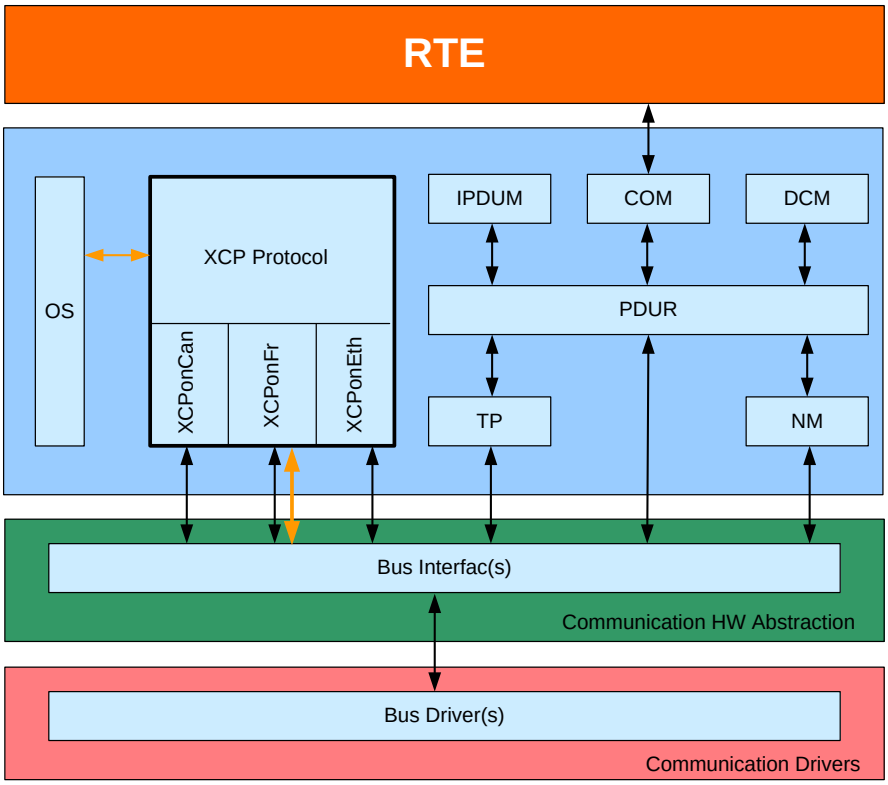

Figure 2: Description
Black arrows: Data Path (Signals/Pdus)
Orange arrows: Control Path (FlexRay Interface)

[SWS_Xcp_00701]⌜
The AUTOSAR XCP Module be located above the bus specific Interfaces in case of FlexRay and Can. In case of Ethernet, the AUTOSAR XCP module shall be located above the Socket Adaptor.⌟(SRS_Xcp_29001)

[SWS_Xcp_00702]⌜
For transmitting and receiving of XCP messages, unique PDU-IDs shall be used. ⌟(SRS_Xcp_29003)

[SWS_Xcp_00703]⌜
The AUTOSAR XCP Module shall support the ASAM XCP Specification Version 1.1.⌟(SRS_Xcp_29004)

[SWS_Xcp_00705]⌜

The AUTOSAR XCP Module shall support the basic feature “Synchronous data acquisition (measurement) “. Please refer to [13] ⌋(SRS_Xcp_29009)

[SWS_Xcp_00706]⌈
The AUTOSAR XCP Module shall support the feature “Dynamic DAQ Configuration”. according to [13] ⌋(SRS_Xcp_29013)

[SWS_Xcp_00707]
⌈The AUTOSAR XCP Module shall support the basic feature “Synchronous data stimulation” according to [13]⌋(SRS_Xcp_29010)

[SWS_Xcp_00708]⌈
The AUTOSAR XCP Module shall support the basic feature “Online memory calibration (read / write access) “, according to [13]⌋()

[SWS_Xcp_00709]⌈
The AUTOSAR XCP Module shall support the feature “Timestamped Data Transfer“, according to [13]⌋(SRS_Xcp_29014)

[SWS_Xcp_00768]⌈
The ECU local time shall be derived from the AUTOSAR OS.⌋()

[SWS_Xcp_00711]⌈
The AUTOSAR XCP Module shall support the feature “Block communication mode“, according to [13]⌋(BSW42911)

[SWS_Xcp_00761]⌈
The AUTOSAR XCP Module shall support the feature “Bypassing“, according to [13]⌋(SRS_Xcp_29015)

[SWS_Xcp_00766]⌈
The AUTOSAR XCP Module shall support the feature “Seed & Key” according to [13]⌋(SRS_Xcp_29016)

[SWS_Xcp_00712]⌈
For sending and receiving of calibration data, the sending and receiving APIs specified within the AUTOSAR BSW Bus Interfaces (FlexRay Interface, CAN Interface, TCP/IP Socket Adaptor) shall be used. Please refer to chapter 7.1, 7.2 and 7.3.⌋(SRS_Xcp_29002)

7.1 XCP on CAN

[SWS_Xcp_00713]⌈
The AUTOSAR XCP Module shall support the CAN communications bus according to [14]⌋(SRS_Xcp_29005)

[SWS_Xcp_00714]

「XCP data sent and received via CAN, the PDUs have to be transmitted and received using the transmitting and receive APIs provided by the AUTOSAR CAN Interface, according to [8]」(SRS_Xcp_29002)

[SWS_Xcp_00715]「
For sending and receiving XCP data via CAN, at least two different CAN identifiers have to be configured to be used by XCP.」()

[SWS_Xcp_00716]「
Performance information shall be exchanged between the XCP master and XCP slave using the parameters according to [14]」()

[SWS_Xcp_00718]「
The XCP Module shall support the GET_SLAVE_ID command according to [14]」()

7.2 XCP on FlexRay

[SWS_Xcp_00719]「
The AUTOSAR XCP Module shall support the FlexRay communications bus according to [16]」(SRS_Xcp_29006)

[SWS_Xcp_00720]「
XCP data sent and received via FlexRay, the PDUs have to be transmitted and received using the transmit and receive APIs provided by the AUTOSAR FlexRay Interface according to [7].」(SRS_Xcp_29002)

[SWS_Xcp_00721]「
All XCP on FlexRay LPDUs always are event driven. Please refer to Chapter 1.1.2 “FlexRay Frame Type” of [16]」()

[SWS_Xcp_00722]「
The hardware buffers (of the FlexRay Communication Controller) XCP uses for data transmission and reception are assigned exclusively to the XCP module.」()

Note:
This restriction prevents disturbances of ongoing FlexRay communication.

[SWS_Xcp_00723]「
The usage of FlexRay Communication Controller’s hardware buffers shall be configured by the corresponding parameters according to [16]」()

[SWS_Xcp_00724]⌈
The FlexRay PDU length used by the AUTOSAR XCP module shall be set using the corresponding parameters according to [16]⌋()

[SWS_Xcp_00725]⌈
LPDU_IDs which shall be routed to the AUTOSAR XCP module (using the AUTOSAR Bus Interface) have to be defined by the system designer. ⌋()

[SWS_Xcp_00726]⌈
The ASAM MCD 2MC description file (i.e. A2L file) describes to which extent the XCP-dedicated buffers of a specific slave can be configured for XCP communication. ⌋()

[SWS_Xcp_00728]⌈
The XCP master gets the information about the XCP dedicated FlexRay Communication Controller buffers from the ASAM MCD 2MC description file. ⌋()

[SWS_Xcp_00729]⌈
Limitations due to the usage of multiple XCP slaves on the FlexRay communications bus shall be taken into consideration by the system designer. Please refer to [16].⌋()

[SWS_Xcp_00730]⌈
Depending upon the requirements on sequencing correctness, alignment and net data throughput, different header types are possible. Please refer to Chapter 1.4.1 ”Header” of [16]⌋()

[SWS_Xcp_00731]
⌈For XCP on FlexRay, the Tail consists of a Control Field containing optional FILL bytes according to [16].⌋()

[SWS_Xcp_00732]⌈
The AUTOSAR XCP module shall be able to pack multiple XCP messages into one FlexRay Frame according to [16].⌋()

7.3 XCP on Ethernet

[SWS_Xcp_00733]⌈
The AUTOSAR XCP Module shall support the Ethernet communications bus according to [15] ⌋(SRS_Xcp_29007)

[SWS_Xcp_00734]⌈
XCP data sent and received via Ethernet, the PDUs have to be transmitted and received using the transmitting and receive APIs provided by the AUTOSAR Socket Adaptor according to [9]. ⌋(SRS_Xcp_29002)

[SWS_Xcp_00735]⌈
The AUTOSAR XCP slave connected by Ethernet and TCP/IP or UDP/IP is addressed by its IP Address and Port number. ⌋()

[SWS_Xcp_00736]⌈
The AUTOSAR XCP slave only accepts one connection at the time. ⌋()

[SWS_Xcp_00737]⌈
If the socket is closed while in XCP connected state, the slave device will perform an XCP disconnect, which means that all data acquisition will be stopped. ⌋()

[SWS_Xcp_00738]⌈
The addressing scheme is defined according to [15] ⌋()
.
[SWS_Xcp_00739]⌈
The header and tail of an XCP on Ethernet message have to be set according to [15] ⌋()

[SWS_Xcp_00740]⌈
The upper performance limit depends on the protocol stack of the host system. The corresponding parameters defined according to [15] have to be set. ⌋()

[SWS_Xcp_00710]⌈
The AUTOSAR XCP Module shall support the feature “Interleaved communication mode“, according to according to [13] ⌋(SRS_Xcp_29012)

7.4 Requirements on Debugging

[**SWS_Xcp_00764**]⌈

The internal XCP states shall be available for debugging.⌋()

In general, it is not necessary/intended for AUTOSAR debugging, that SWS documents define specific variables.

7.4.1 General Requirements

[SWS_Xcp_00741]⌈
Link-time and post-build-time configuration data shall be implemented as read-only data structures. Link-time configuration data shall be immediately referenced by the implementation, the start-address of post-build-time configuration data shall be passed during module initialization⌋(SRS_BSW_00344)

[SWS_Xcp_00742]
⌈The XCP module shall support pre-compile time, link-time and post-build-time configuration.⌋(SRS_BSW_00404, SRS_BSW_00345)

7.5 Error classification

[**SWS_Xcp_00763**]
⌈The error values and EventIds are named in capital letters according to the scheme `XCP_E_<NAME>`, where NAME describes the error/EventId and may consist of several words separated by underscores.⌋(SRS_BSW_00327)

Type or error	*Relevance*	*Related error code*	*Value [hex]*
Invalid pointer	Development	`XCP_E_INV_POINTER`	0x01
Module not initialized	Development	`XCP_E_NOT_INITIALIZED`	0x02
API call with wrong PDU ID	Development	`XCP_E_INVALID_PDUID`	0x03
Initialization of XCP failed	Development	`XCP_E_INIT_FAILED`	0x04
Null pointer has been passed as an argument	Development	`XCP_E_NULL_POINTER`	0x12

7.6 Error detection

For details refer to the chapter 7.3 “Error Detection” in *SWS_BSWGeneral.*

7.7 Error notification

For details refer to the chapter 7.4 “Error notification” in *SWS_BSWGeneral.*

7.8 Version checking

For details refer to the chapter 5.1.8 “Version Check” in *SWS_BSWGeneral.*

8 API specification

8.1 Imported types

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

[**SWS_Xcp_00801**]⌈

Module	*Imported Type*
ComStack_Types	NetworkHandleType
	PduIdType
	PduInfoType
Dem	Dem_EventIdType
	Dem_EventStatusType
Os	CounterType
	StatusType
	TickRefType
Std_Types	Std_ReturnType
	Std_VersionInfoType

⌋()

8.2 Type definitions

8.2.1 Xcp_ConfigType

[SWS_Xcp_00845]⌈

Name:	`Xcp_ConfigType`	
Type:	`Structure`	
Range:	`implementation specific`	The content of the initialization data structure is implementation specific
Description:	This is the type of the data structure containing the initialization data for XCP.	

⌋()

8.2.2 Xcp_Transmission Mode Type

[SWS_Xcp_00846]⌈

Name:	`Xcp_TransmissionModeType`	
Type:	`Enumeration`	
Range:	`XCP_TX_OFF`	Transmission Disabled
	`XCP_TX_ON`	Transmission Enabled
Description:	Handles the enabling and disabling of the transmission mode	

⌋()

8.3 Function definitions

This is a list of functions provided for upper layer modules.

8.3.1 Xcp_Init

[**SWS_Xcp_00803**] ⌈

Service name:	Xcp_Init	

Syntax:	`void Xcp_Init(`
	`  const Xcp_ConfigType* Xcp_ConfigPtr`
	`)`	

Service ID[hex]:	0x00	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	Xcp_ConfigPtr	Pointer to a selected configuration structure
Parameters (inout):	None	
Parameters (out):	None	
Return value:	void	--
Description:	This service initializes interfaces and variables of the AUTOSAR XCP layer.	

⌋(SRS_BSW_00405, SRS_BSW_00101, SRS_BSW_00358, SRS_BSW_00414, SRS_Xcp_29017)

[**SWS_Xcp_00802**] ⌈The function `Xcp_Init` shall internally store the configuration address to enable subsequent API calls to access the configuration⌋()

[**SWS_Xcp_00834**] ⌈
If development error detection for the XCP module is enabled (`XcpDevErrorDetect` is `ON`): the function `Xcp_Init` shall check the parameter `Xcp_ConfigType` for not being a NULL pointer (`NULL_PTR`). If `Xcp_ConfigType` is a NULL pointer, the function `Xcp_Init` shall raise the development error `XCP_E_INV_POINTER` and return.⌋()

8.3.2 Xcp_GetVersionInfo

[SWS_Xcp_00807] ⌈

Service name:	Xcp_GetVersionInfo	
Syntax:	`void Xcp_GetVersionInfo( Std_VersionInfoType* versioninfo )`	
Service ID[hex]:	0x01	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	versioninfo	Pointer to where to store the version information of this module.
Return value:	void	--
Description:	Returns the version information of this module.	

⌋(SRS_BSW_00402, SRS_BSW_00407, SRS_BSW_00411, SRS_BSW_00374, SRS_BSW_00379, SRS_BSW_00003, SRS_BSW_00318)

[**SWS_Xcp_00825**] ⌈
If development error detection for the Xcp module is enabled, then the function `Xcp_GetVersionInfo` shall check whether the parameter VersioninfoPtr is a NULL pointer (`NULL_PTR`). If VersioninfoPtr is a NULL pointer, then the function `Xcp_GetVersionInfo` shall raise the development error `XCP_E_INV_POINTER` and return.⌋()

8.4 Call-back notifications

[**SWS_Xcp_00836**] ⌈
This is a list of functions provided for other modules. The function prototypes of the callback functions shall be provided in the file Xcp_Cbk.h⌋()

8.4.1 Xcp_<module>RxIndication

[**SWS_Xcp_00813**] ⌈

Service name:	Xcp_<module>RxIndication	

Syntax:	`void Xcp_<module>RxIndication(`
	`  PduIdType XcpRxPduId,`
	`  const PduInfoType* XcpRxPduPtr`
	`)`	

Service ID[hex]:	0x03	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different XcpRxPduIds, non reentrant for the same XcpRxPduId	
Parameters (in):	XcpRxPduId	PDU-ID that has been received
	XcpRxPduPtr	Pointer to SDU (Buffer of received payload)
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	This function is called by the lower layers (i.e. FlexRay Interface, TTCAN Interface and Socket Adaptor or CDD) when an AUTOSAR XCP PDU has been received	

⌋()

The callback function `Xcp_<module>RxIndication` is called by the Bus Interfaces, Ethernet Socket Adaptor or CDD and is implemented by the Xcp module.

[SWS_Xcp_00847] ⌈
The callback function `Xcp_<module>RxIndication` shall inform the DET, if development error detection is enabled (**`XCP_DEV_ERROR_DETECT`** is set to TRUE) and if function call has failed because of the following reasons:

- Xcp was not initialized (`XCP_E_NO_INIT`)
- XcpRxPduPtr equals NULL_PTR (`XCP_E_NULL_POINTER`)
- Invalid PDUID (`XCP_E_INVALID_PDUID`)⌋()

The function `Xcp_<module>RxIndication` shall be called by the Xcp module's environment in an interrupt context.

8.4.2 Xcp_<module>TxConfirmation

[**SWS_Xcp_00814**] ⌜

Service name:	Xcp_<module>TxConfirmation	

Syntax:	`void Xcp_<module>TxConfirmation(`
	`  PduIdType XcpTxPduId`
	`)`	

Service ID[hex]:	0x02	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different XcpTxPduIds, non reentrant for the same XcpTxPduId	
Parameters (in):	XcpTxPduId	PDU-ID that has been transmitted
Parameters (inout):	None	
Parameters (out):	None	
Return value:	void	--
Description:	This function is called by the lower layers (i.e. FlexRay Interface, TTCAN Interface and Socket Adaptor or CDD) when an AUTOSAR XCP PDU has been transmitted	

⌟()

Note:
The callback function `Xcp_<module>TxConfirmation` is called by the Bus Interfaces, Ethernet Socket Adaptor or CDD and is implemented by the Xcp module.

[**SWS_Xcp_00840**] ⌜
If development error detection for the XCP module is enabled: if the function `Xcp_<module>TxConfirmation` is called before the XCP was initialized successfully, the function `Xcp_<module>TxConfirmation` shall raise the development error `XCP_E_NOT_INITIALIZED` and return.⌟()

[**SWS_Xcp_00841**] ⌜
Caveats of Xcp_<module>TxConfirmation:

- The call context is either on interrupt level (interrupt mode) or on task level
- The Xcp module is initialized correctly.

⌟()

8.4.3 Xcp_<Lo>TriggerTransmit

[**SWS_Xcp_00835**] ⌜

Service name:	Xcp_<Lo>TriggerTransmit

Syntax:	`Std_ReturnType Xcp_<Lo>TriggerTransmit(`
	`  PduIdType TxPduId,`
	`  PduInfoType* PduInfoPtr`
	`)`

Service ID[hex]:	0x41
Sync/Async:	Synchronous
Reentrancy:	Reentrant for different PduIds. Non reentrant for the same PduId.

Parameters (in):	TxPduId	ID of the SDU that is requested to be transmitted.
	PduInfoPtr	Contains a pointer to a buffer (SduDataPtr) to where the SDU data shall be copied. On return, the service will indicate the length of the copied SDU data in SduLength.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: SDU has been copied and SduLength indicates the number of copied bytes. E_NOT_OK: No SDU data has been copied. PduInfoPtr must not be used since it may contain a NULL pointer or point to invalid data.
Description:	Within this API, the upper layer module (called module) shall copy its data into the buffer provided by PduInfoPtr->SduDataPtr and update the length of the actual copied data in PduInfoPtr->SduLength.	

」()

Note:
The callback function `Xcp_<module>TriggerTransmit` is called by the Bus Interfaces, Ethernet Socket Adaptor or CDD and is implemented by the Xcp module.

[**SWS_Xcp_00842**] 「
If development error detection for the XCP module is enabled: if the function `Xcp_<module>TriggerTransmit` is called before the XCP was initialized successfully, the function `Xcp_<module>TriggerTransmit` shall raise the development error `XCP_E_NOT_INITIALIZED` and return `E_NOT_OK`.」()

[**SWS_Xcp_00843**] 「
Caveats of Xcp_<module>TriggerTransmit:

- The call context is either on interrupt level (interrupt mode) or on task level
- The Xcp module is initialized correctly.」()

8.4.4 Xcp_SetTransmissionMode

[SWS_Xcp_00844]「

Service name:	Xcp_SetTransmissionMode	

Syntax:	`void Xcp_SetTransmissionMode(`
	`  NetworkHandleType Channel,`
	`  Xcp_TransmissionModeType Mode`
	`)`	

Service ID[hex]:	0x05	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	Channel	The Network channel for the used bus communication
	Mode	Enabled or disabled Transmission mode Parameters
Parameters (inout):	None	
Parameters (out):	None	

Return value:	None
Description:	This API is used to turn on and off of the TX capabilities of used communication bus channel in XCP module.

⌋()

[SWS_Xcp_00848] ⌈

The XCP module shall provide this service only if `XCP_SUPPRESS_TX_SUPPORT` (see `ECUC_Xcp_00169`) equals `TRUE`.⌋()

[SWS_Xcp_00849]⌈

If `Xcp_SetTransmissionMode(Channel, Mode)` is called and parameter `Mode` equals `XCP_TX_OFF`, all TxPDUs which are assigned to `Channel` shall not be transmitted. ⌋()

Note: It could be derived from <Bus>If configuration and the global PDU parameter, to which specific communication channel the PDU is assigned to.

[SWS_Xcp_00850] ⌈

If `Xcp_SetTransmissionMode(Channel, Mode)` is called and parameter `Mode` equals `XCP_TX_ON`, all TxPDUs which are assigned to `Channel` shall be able to be transmitted. ⌋()

8.5 Scheduled functions

The functions are called directly by Basic Software Scheduler. The following functions shall have no return value and no parameter. All functions shall be non reentrant.

8.5.1 Xcp_MainFunction

[SWS_Xcp_00823] ⌈

Service name:	Xcp_MainFunction

Syntax:	`void Xcp_MainFunction(`
	`  void`
	`)`

Service ID[hex]:	0x04
Description:	Scheduled function of the XCP module

Service name:	Xcp_MainFunction

Syntax:	`void Xcp_MainFunction(`
	`  void`
	`)`

Service ID[hex]:	0x04
Description:	Scheduled function of the XCP module

⌋(SRS_BSW_00424, SRS_BSW_00433, SRS_BSW_00373)

[**SWS_Xcp_00824**] ⌈

The XCP Main Function shall be called cyclically.⌋()

8.6 Expected Interfaces

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

8.6.1 Mandatory Interfaces

API function	***Description***
<UL_RxIndication>	This API service of an upper layer BSW module <UL> (e.g. PduR, FrTp, FrNm, Xcp) is called by the FlexRay Interface to indicate to this upper layer BSW module that the PDU with index FrIf_RxPduId has been received via the FlexRay Communication System.
<UL_TxConfirmation>	This API service of an upper layer BSW module <UL> (e.g. PduR, FrTp, FrNm, Xcp) is called by the FlexRay Interface to confirm to this upper layer BSW module that the PDU with index FrIf_TxPduId has been transmitted via the FlexRay Communication System.
<User_RxIndication>	This service indicates a successful reception of a received Message to the corresponding UL module. This service provides the syntax with a PDU Info pointer.
<User_TxConfirmation>	This service confirms a previous successfully processed CAN transmit request.
CanIf_Transmit	This service initiates a request for transmission of the CAN L-PDU specified by the CanTxSduId and CAN related data in the L-SDU structure.
Dem_ReportErrorStatus	Queues the reported events from the BSW modules (API is only used by BSW modules). The interface has an asynchronous behavior, because the processing of the event is done within the Dem main function. OBD Events Suppression shall be ignored for this computation.
FrIf_Transmit	Requests the sending of a PDU.
SoAd_IfTransmit	This service initiates a request for transmission of the L-PDU specified by the SoAdSrcPduId. The corresponding socket has to be resolved by the SoAdSrcPduId. This call is used to mimic the call to an IF in AUTOSAR. Development errors: Invalid values of SoAdSrcPduId or SoAdSrcPduInfoPtr will be reported to the development error tracer (SOAD_E_INVALID_TXPDUID or SOAD_E_PARAM_POINTER).

8.6.2 Optional Interfaces

[**SWS_Xcp_00832**] ⌈

API function	***Description***
Det_ReportError	Service to report development errors.
GetCounterValue	This service reads the current count value of a counter (returning either

	the hardware timer ticks if counter is driven by hardware or the software ticks when user drives counter).
GetElapsedValue	This service gets the number of ticks between the current tick value and a previously read tick value.

⌋()

[**SWS_Xcp_00833**] ⌈

API function	***Description***
FrIf_ReconfigLPdu	API can be used for buffer reconfiguration of the FlexRay CC
FrIf_DisableLPdu	API can be used to disable the transmission or reception of LPdus

⌋()

8.6.3 Configurable interfaces

In this chapter, all interfaces are listed where the target function could be configured. The target function is usually a call-back function. The names of these kind of interfaces is not fixed because they are configurable.

The XCP module offers configurable interfaces to be used by Complex Driver(s).

[**SWS_Xcp_00831**] ⌈

API function	***Description***
<Cdd_Transmit>	Requests the sending of a PDU via Complex Driver
<Xcp_CddTxConfirmation>	API confirming the successful transmission of the PDU
<Xcp_CddRxIndication>	This API service called by the AUTOSAR Cdd indicates a successful reception of an L-PDU.
⌋()	

9 Sequence diagrams

9.1 XCP on FlexRay

9.1.1 Xcp on FlexRay Transmit

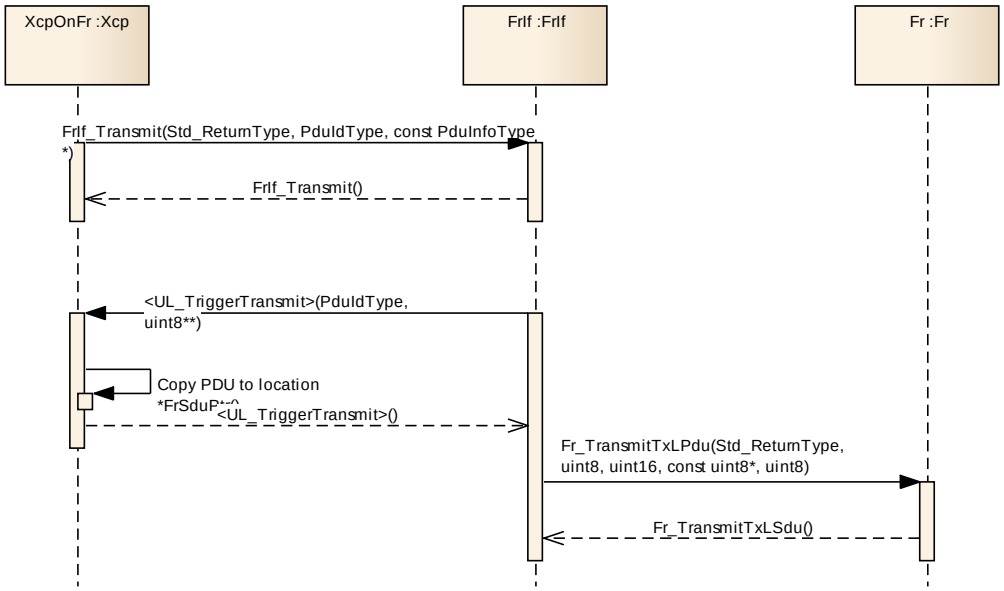

Figure 3: Xcp On FlexRay Transmit

9.1.2 Xcp on FlexRay Receive Indication

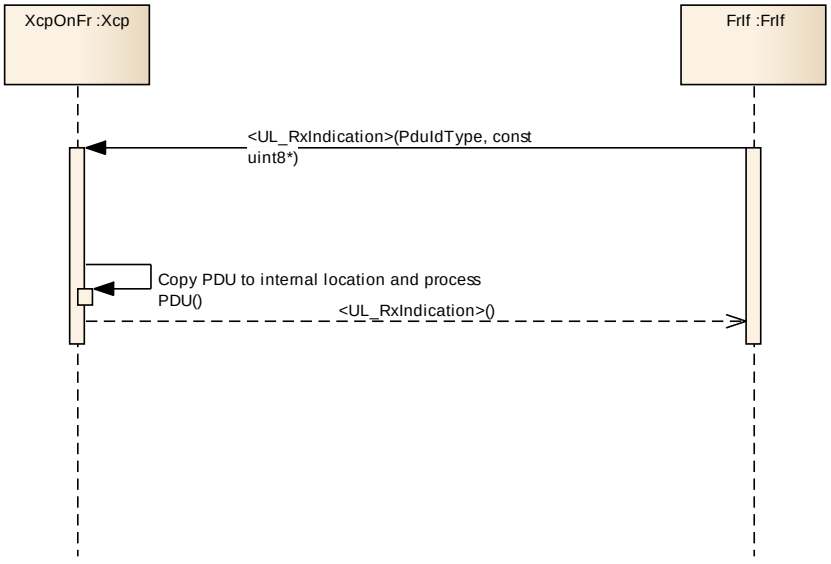

Figure 4: Xcp on FlexRay Receive Indication

9.2 XCP on CAN

9.2.1 Xcp on CAN Transmit

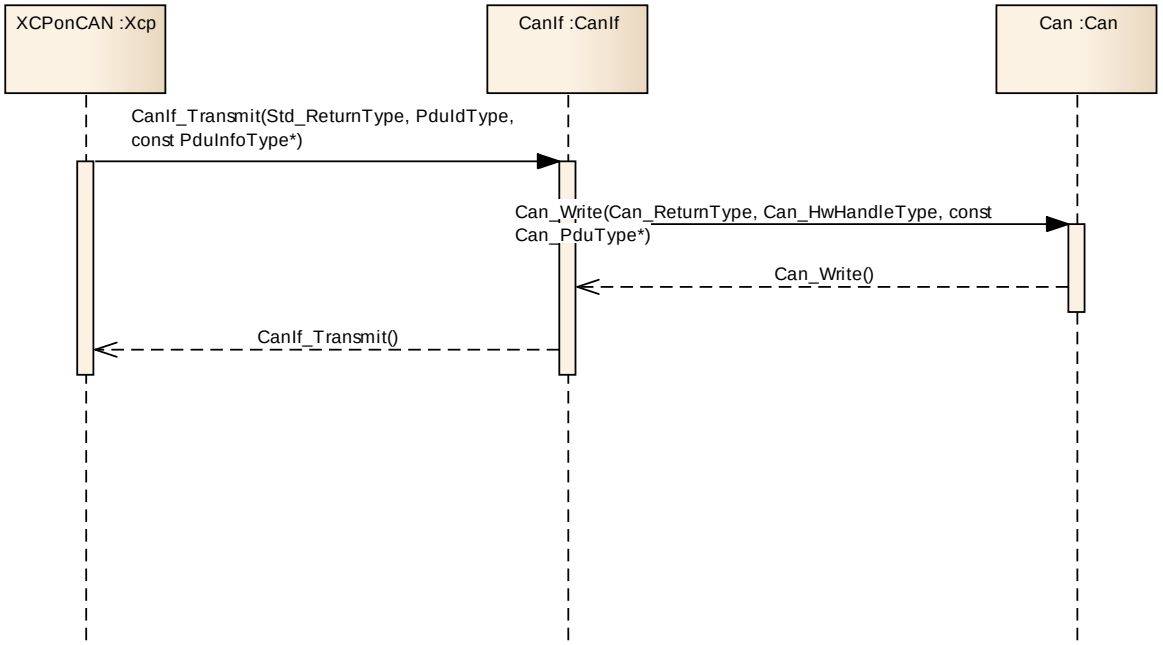

Figure 5: Xcp on Can Transmit

9.2.2 Xcp on CAN Transmit Confirmation

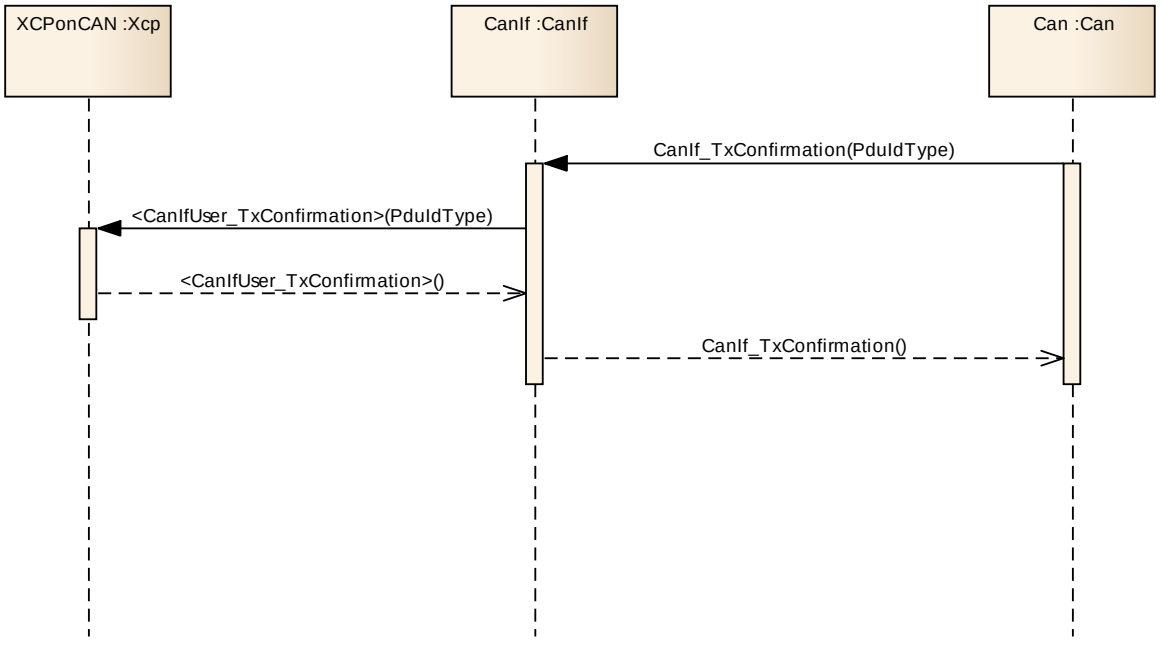

Figure 6: Xcp on CAN Transmit Confirmation

9.2.3 Xcp on CAN Receive Indication

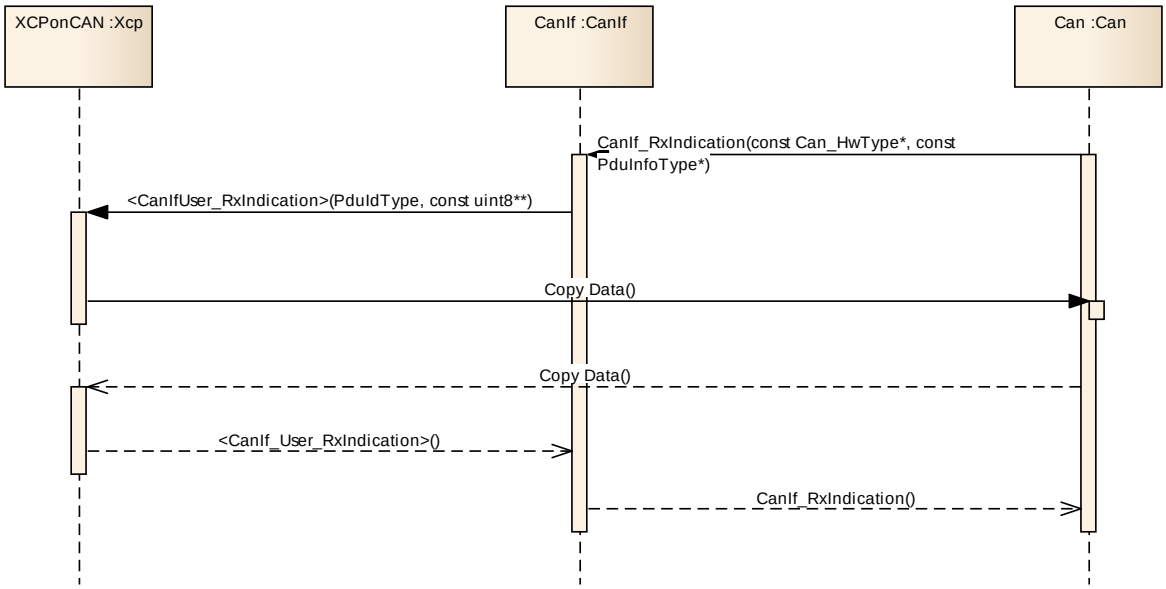

Figure 7: Xcp on CAN Receive Indication

9.3 XCP on Ethernet

9.3.1 Xcp on Ethernet Receive Indication

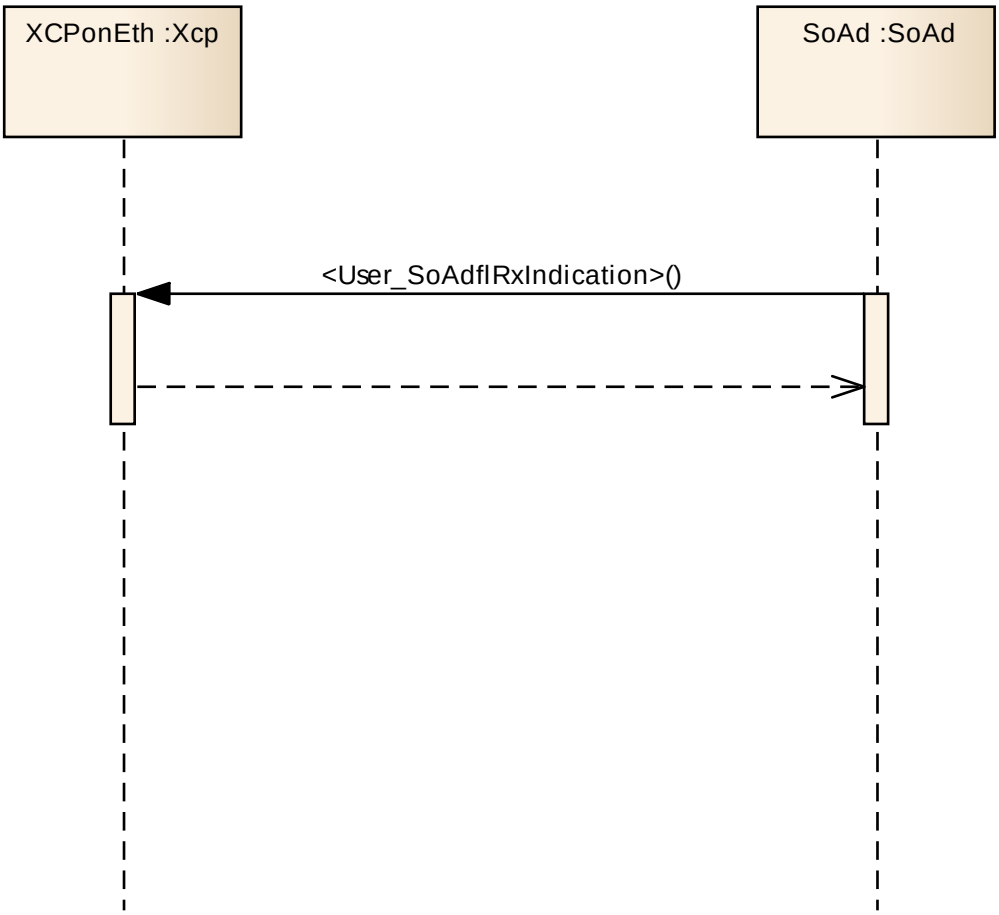

Figure 8: Xcp on Ethernet Receive Indication

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 XCP.

Chapter 10.3 specifies published information of the module XCP.

10.1 How to read this chapter

For details refer to the chapter 10.1 “Introduction to configuration specification” in *SWS_BSWGeneral.*

10.2 Containers and configuration parameters

The following chapters summarize all configuration parameters. The detailed meanings of the parameters describe Chapters 7 and Chapter 8.

[**SWS_Xcp_00102**] ⌈
The listed configuration items can be derived from a network description database, which is based on the EcuConfigurationTemplate. The configuration tool shall extract all information to configure the XCP.⌋(SRS_BSW_00159)

[**SWS_XCP_00103**] ⌈
The configuration tool must check the consistency of the configuration at configuration time.⌋(SRS_BSW_00167)

[**SWS_Xcp_00104**] ⌈
Configuration rules and constraints for plausibility checks shall be performed during configuration time, wherever possible.⌋(SRS_BSW_00167)

[**SWS_Xcp_00105**] ⌈
These dependencies between FlexRay Interface and FlexRay Driver configuration must be provided at configuration time by the configuration tools. ⌋ (SRS_BSW_00167)

10.2.1 Variants

VARIANT-POST-BUILD: All configuration parameters in container 'XcpGeneral' shall be configurable at pre-compile time. All other configuration parameters shall be configurable at post-build-time.

Use case: Object code delivery, selectable configuration

VARIANT-PRE-COMPILE: All configuration parameters shall be configurable at pre-compile time.

Use case: Execution time optimizations

VARIANT-LINK-TIME: It shall include all configuration options of the variant VARIANT-PRE-COMPILE. Additionally all parameters that are marked as link-time configurable with "VARIANT-LINK-TIME" shall be configurable at link time for example by linking a special configured parameter object file.

10.2.2 Xcp

Module Name	*Xcp*
Module Description	Configuration of the XCP module

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
XcpConfig	1	--
XcpGeneral	1	This container contains the general configuration parameters of the XCP.

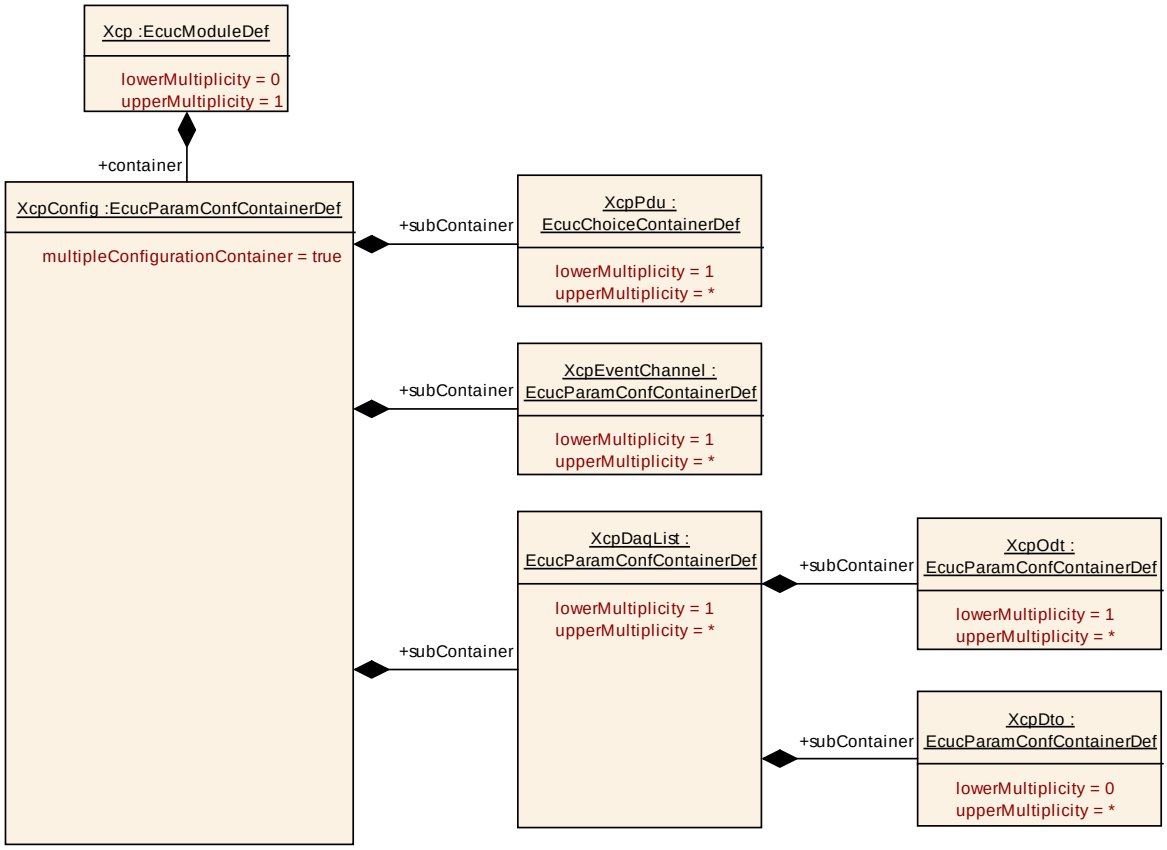

Figure 9: Diagram

10.2.3 XcpGeneral

SWS Item	**ECUC_Xcp_00001 :**
Container Name	XcpGeneral
Description	This container contains the general configuration parameters of the XCP.
Configuration Parameters	

SWS Item	**ECUC_Xcp_00164 :**		
Name	XcpDaqConfigType		
Description	Sets the DAQ_CONFIG_TYPE bit within the DAQ_PROPERTIES parameter to "static" or to "dynamic". If DAQ_STATIC is selected, the DAQ_CONFIG_TYPE bit is set to "0". If DAQ_DYNAMIC is selected, the DAQ_CONFIG_TYPE bit is set to "1".		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	DAQ_DYNAMIC	If DAQ_DYNAMIC is selected, the DAQ_CONFIG_TYPE bit is set to '1'	
	DAQ_STATIC	If DAQ_STATIC is selected, the DAQ_CONFIG_TYPE bit is set to '0'	
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU dependency: If DAQ_CONFIG_TYPE = dynamic, MAX_DAQ equals MIN_DAQ+DAQ_COUNT.		

SWS Item	ECUC_Xcp_00012 :		
Name	XcpDaqCount {XCP_DAQ_COUNT}		
Description	Indicates the number of DAQ lists for dynamic configuration.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 65535		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU dependency: This parameter is available only if XcpDaqConfigType is set to "1" i.e DAQ_DYNAMIC		

SWS Item	ECUC_Xcp_00003 :		
Name	XcpDevErrorDetect {XCP_DEV_ERROR_DETECT}		
Description	Switches the Development Error Detection and Notification on or off. TRUE: Development Error Detection and Notification on FALSE: Development Error Detection and Notification off		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	ECUC_Xcp_00170 :		
Name	XcpIdentificationFieldType {XCP_IDENTIFICATION_FIELD_TYPE}		
Description	Type of Identification Field the slave will use when transferring DAQ Packets to the master. The master has to use the same Type of Identification Field when transferring STIM Packets to the slave.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	ABSOLUTE		Absolute ODT number
	RELATIVE_BYTE		Relative ODT number, absolute DAQ list number (BYTE)
	RELATIVE_WORD		Relative ODT number, absolute DAQ list number (WORD)
	RELATIVE_WORD_ALIGNED		Relative ODT number, absolute DAQ list number (WORD, aligned).
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	ECUC_Xcp_00014 :		
Name	XcpMainFunctionPeriod {XCP_MAIN_FUNCTION_PERIOD}		
Description	The XCP does not require this information but the BSW scheduler, which invokes the main function, needs it in order to plan its tasks.		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	0 .. INF		
Default value	--		

ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Xcp_00004 :**		
Name	XcpMaxCto {XCP_MAX_CTO}		
Description	MAX_CTO shows the maximum length of a CTO packet in bytes.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	8 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Xcp_00005 :**		
Name	XcpMaxDto {XCP_MAX_DTO}		
Description	MAX_DTO shows the maximum length of a DTO packet in bytes.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	8 .. 65535		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Xcp_00011 :**		
Name	XcpMaxEventChannel {XCP_MAX_EVENT_CHANNEL}		
Description	--		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 65535		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_Xcp_00013 :**		
Name	XcpMinDaq {XCP_MIN_DAQ}		
Description	Indicates the number of predefined, read only DAQ lists on the XCP slave.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_Xcp_00054 :**
Name	XcpOdtCount {XCP_ODT_COUNT}

Description	This parameter indicates the amount of ODTs of a DAQ list using dynamic DAQ list configuration.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 252		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU dependency: This parameter is available only if XcpDaqConfigType is set to "1" i.e DAQ_DYNAMIC		

SWS Item	**ECUC_Xcp_00059 :**		
Name	XcpOdtEntriesCount {XCP_ODT_ENTRIES_COUNT}		
Description	Indicates the amount of entries into an ODT using dynamic DAQ list configuration.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU dependency: This parameter is available only if XcpDaqConfigType is set to "1" i.e DAQ_DYNAMIC		

SWS Item	**ECUC_Xcp_00177 :**		
Name	XcpOdtEntrySizeDaq {XCP_ODT_ENTRY_SIZE_DAQ}		
Description	Indicates the size of an element described by an ODT entry to the DaqListType for a DAQ.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_Xcp_00178 :**		
Name	XcpOdtEntrySizeStim {XCP_ODT_ENTRY_SIZE_STIM}		
Description	Indicates the size of an element described by an ODT entry to the DaqListType for a stim.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_Xcp_00006 :**
Name	XcpOnCanEnabled {XCP_ON_CAN_ENABLED}

Description	Enabling of XCPonCAN functionality		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Xcp_00009 :**		
Name	XcpOnCddEnabled {XCP_ON_CDD_ENABLED}		
Description	Enabling of XCPonCdd functionality		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Xcp_00008 :**		
Name	XcpOnEthernetEnabled {XCP_ON_ETHERNET_ENABLED}		
Description	Enabling of XCPonEthernet functionality		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Xcp_00007 :**		
Name	XcpOnFlexRayEnabled {XCP_ON_FLEXRAY_ENABLED}		
Description	Enabling of XCPonFlexRay functionality		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Xcp_00169 :**		
Name	XcpPrescalerSupported {XCP_PRESCALER_SUPPORTED}		
Description	This parameter enables and disables the support for Prescaler support. True is Enabled, False is disabled		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Xcp_00176 :**

Name	XcpSuppressTxSupport {XCP_SUPPRESS_TX_SUPPORT}		
Description	Switches the support of suppressing transmission of PDUs per communication channel on or off. TRUE: Suppressing of TxPDUs supported FALSE: Suppressing of TxPDUs not supported		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Xcp_00167 :**		
Name	XcpTimestampTicks {XCP_TIMESTAMP_TICKS}		
Description	This parameter defines the timestamp that will increment based TIMESTAMP_TICKS per unit and wrap around if an overflow occurs.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 65535		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Xcp_00166 :**		
Name	XcpTimestampType {XCP_TIMESTAMP_TYPE}		
Description	This parameter indicates the number of bytes used for the timestamp field. In case No_TIME_STAMP is selected the timestamp field is not available.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	FOUR_BYTE	timestamp field has the size of four byte.	
	NO_TIME_STAMP	timestamp field is not available.	
	ONE_BYTE	timestamp field has the size of one byte.	
	TWO_BYTE	timestamp field has the size of two byte.	
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Xcp_00168 :**	
Name	XcpTimestampUnit {XCP_TIMESTAMP_UNIT}	
Description	This parameter indicates the resolution of the data acquisition clock of the slave when transferring data to master.	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	TIMESTAMP_UNIT_100MS	Unit is 100 millisecond.
	TIMESTAMP_UNIT_100NS	Unit is 100 nanosecond.
	TIMESTAMP_UNIT_100PS	Unit is 100 picosecond.
	TIMESTAMP_UNIT_100US	Unit is 100 microsecond.
	TIMESTAMP_UNIT_10MS	Unit is 10 millisecond.
	TIMESTAMP_UNIT_10NS	Unit is 10 nanosecond.

	TIMESTAMP_UNIT_10PS	Unit is 10 picosecond.	
	TIMESTAMP_UNIT_10US	Unit is 10 microsecond.	
	TIMESTAMP_UNIT_1MS	Unit is 1 millisecond.	
	TIMESTAMP_UNIT_1NS	Unit is 1 nonasecond.	
	TIMESTAMP_UNIT_1PS	Unit is 1 picosecond.	
	TIMESTAMP_UNIT_1S	Unit is 1 second.	
	TIMESTAMP_UNIT_1US	Unit is 1 microsecond.	
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Xcp_00002 :**		
Name	XcpVersionInfoApi {XCP_VERSION_INFO_API}		
Description	Enables/disables the existence of the XCP_GetVersionInfo() API service. TRUE: XCP_GetVersionInfo() API service exists FALSE: XCP_GetVersionInfo() API service does not exist		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Xcp_00162 :**		
Name	XcpCounterRef {XCP_COUNTER_REF}		
Description	This parameter contains a reference to the counter, which is used by XCP.		
Multiplicity	1		
Type	Reference to [OsCounter]		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers

10.2.4 XcpConfig

SWS Item	**ECUC_Xcp_00020 :**
Container Name	XcpConfig [Multi Config Container]
Description	--
Configuration Parameters	

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
XcpDaqList	1..*	This container contains the configuration of the DAQs.
XcpEventChannel	1..*	This container contains the configuration of event channels on the XCP slave.
XcpPdu	1..*	Contains PDU information. A PDU may be either a transmission PDU or a reception PDU.

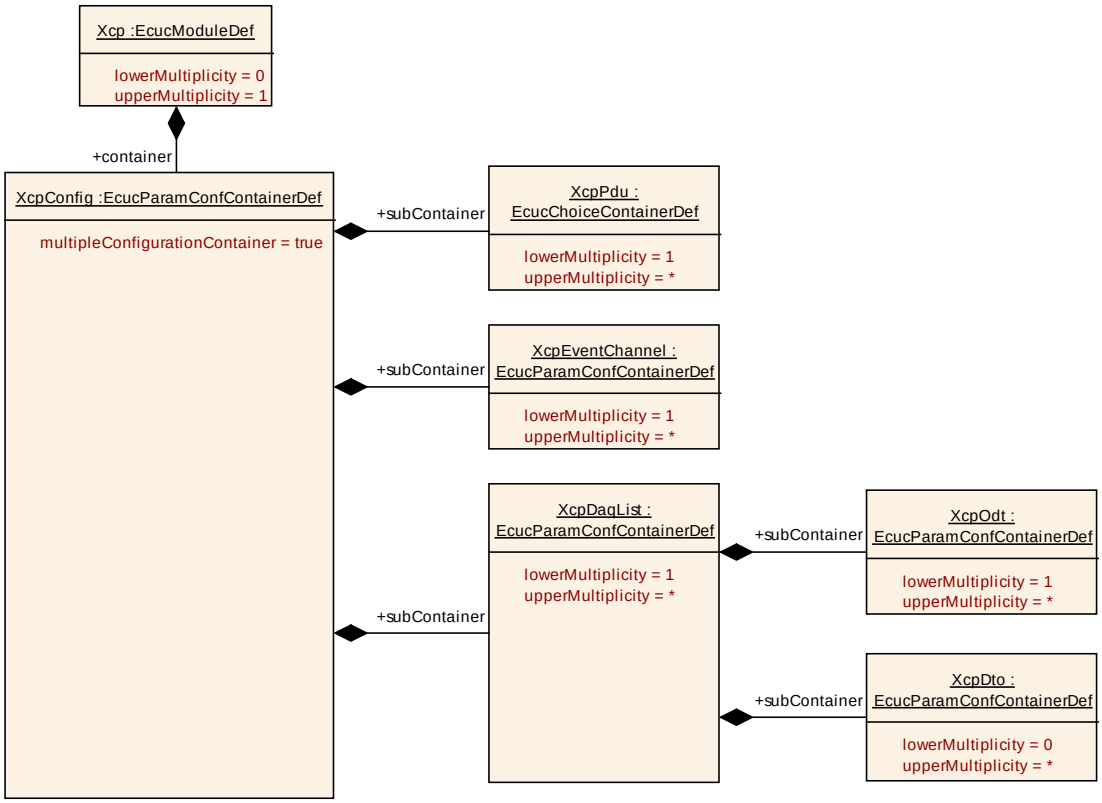

Figure 10: Diagram XcpConfig (1)

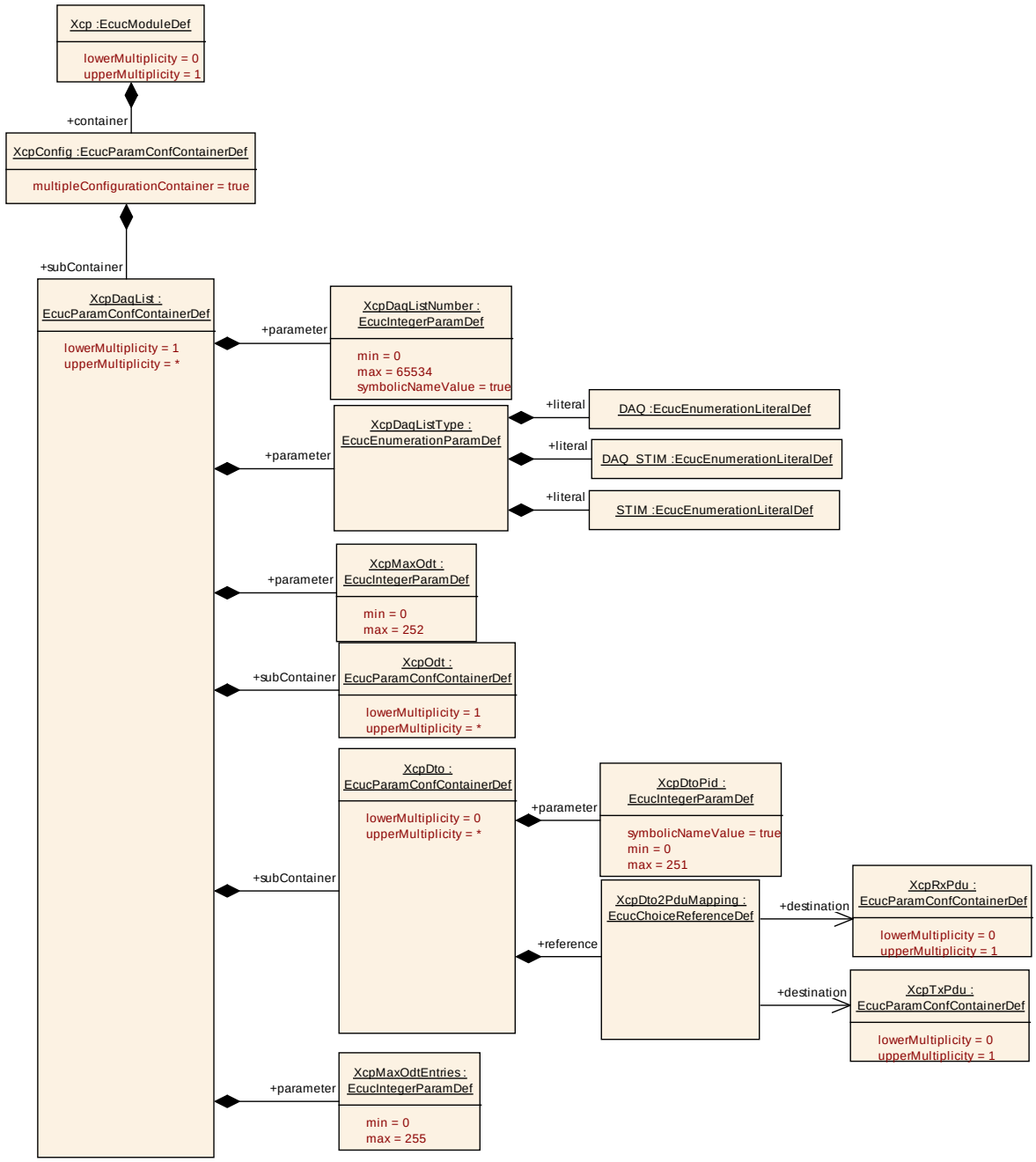

Figure 11: Diagram XcpConfig (2)

10.2.5 XcpDaqList

SWS Item	**ECUC_Xcp_00050 :**
Container Name	XcpDaqList{XCP_DAQ_LIST}
Description	This container contains the configuration of the DAQs.
Configuration Parameters	

SWS Item	**ECUC_Xcp_00051 :**
Name	XcpDaqListNumber {XCP_DAQ_NUMBER}
Description	Index number of the DAQ list

Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 65534		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_Xcp_00052 :**		
Name	XcpDaqListType {XCP_DAQ_LIST_TYPE}		
Description	This indicates whether this DAQ list represents a DAQ or a STIM.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	DAQ	This DAQ list is a DAQ.	
	DAQ_STIM	This DAQ list can be DAQ or STIM.	
	STIM	This DAQ list is a STIM.	
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_Xcp_00053 :**		
Name	XcpMaxOdt {XCP_MAX_ODT}		
Description	MAX_ODT indicates the maximum amount of ODTs in this DAQ list (STATIC configuration)		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 252		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU dependency: only available if XcpDaqConfigType is "DAQ_STATIC" (bit set to '0')		

SWS Item	**ECUC_Xcp_00058 :**		
Name	XcpMaxOdtEntries {XCP_MAX_ODT_ENTRIES}		
Description	This parameter indicates the maximum amount of entries in an ODT of this DAQ list (STATIC configuration).		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU dependency: only available if XcpDaqConfigType is "DAQ_STATIC" (bit set to '0')		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
XcpDto	0..*	This container collects data transfer object specific parameters for the DAQ list.

XcpOdt	1..*	This container contains ODT-specific parameter for the DAQ list.

10.2.6 XcpDto

SWS Item	**ECUC_Xcp_00065 :**
Container Name	XcpDto
Description	This container collects data transfer object specific parameters for the DAQ list.
Configuration Parameters	

SWS Item	**ECUC_Xcp_00066 :**		
Name	XcpDtoPid		
Description	Packet identifier (PID) of the DTO that identifies the ODT the content of the DTO.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 251		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_Xcp_00067 :**		
Name	XcpDto2PduMapping		
Description	This reference specifies the mapping of the DTO to the PDUs from the lower-layer interfaces (CanIf, FrIf, SoAd and Cdd). A reference to a XcpRxPdu is only feasible if the the DaqListType is DAQ_STIM. A reference to a XcpTxPdu is only feasible if the DaqListType is DAQ.		
Multiplicity	1		
Type	Choice reference to [XcpRxPdu , XcpTxPdu]		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

No Included Containers

10.2.7 XcpOdt

SWS Item	**ECUC_Xcp_00055 :**
Container Name	XcpOdt{XCP_ODT}
Description	This container contains ODT-specific parameter for the DAQ list.
Configuration Parameters	

SWS Item	**ECUC_Xcp_00060 :**
Name	XcpOdtEntryMaxSize
Description	This parameter indicates the upper limit for the size of the element described by an ODT entry. Depending on the DaqListType this ODT

	belongs to it describes the limit for a DAQ (MAX_ODT_ENTRY_SIZE_DAQ) or a STIM (MAX_ODT_ENTRY_SIZE_STIM).		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 254		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_Xcp_00057 :**		
Name	XcpOdtNumber {XCP_ODT_NUMBER}		
Description	Index number of this ODT within the DAQ list.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 251		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_Xcp_00056 :**		
Name	XcpOdt2DtoMapping		
Description	This reference maps the ODT to the according DTO in which it will be transmitted.		
Multiplicity	1		
Type	Reference to [XcpDto]		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
XcpOdtEntry	1..*	This container collects all configuration parameters that comprise an ODT entry.

10.2.8 XcpOdtEntry

SWS Item	**ECUC_Xcp_00061 :**
Container Name	XcpOdtEntry{XCP_ODT_ENTRY}
Description	This container collects all configuration parameters that comprise an ODT entry.
Configuration Parameters	

SWS Item	**ECUC_Xcp_00063 :**
Name	XcpOdtEntryAddress
Description	Memory address that the ODT entry is referencing to.
Multiplicity	1
Type	EcucLinkerSymbolDef

Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_Xcp_00179 :**		
Name	XcpOdtEntryBitOffset {XCP_ODT_ENTRY_BIT_OFFSET}		
Description	Represent the bit offset in case of the element represents status bit.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 31		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_Xcp_00064 :**		
Name	XcpOdtEntryLength		
Description	Length of the referenced memory area that is referenced by the ODT entry.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_Xcp_00062 :**		
Name	XcpOdtEntryNumber {XCP_ODT_ENTRY_NUMBER}		
Description	Index number of the ODT entry		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 254		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

No Included Containers

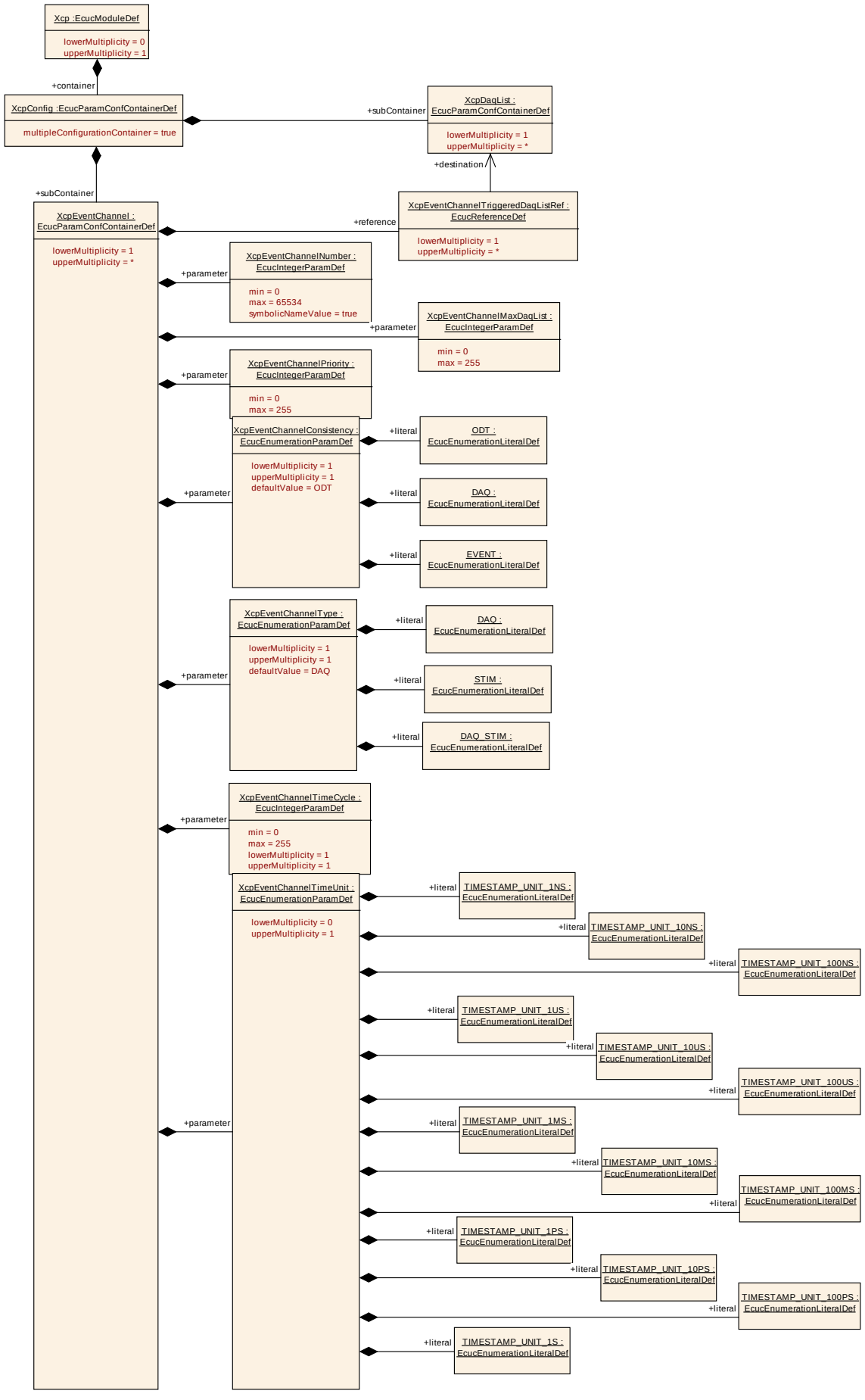
Xcp :EcucModuleDef
lowerMultiplicity = 0
upperMultiplicity = 1
+container
XcpConfig :EcucParamConfContainerDef
multipleConfigurationContainer = true
+subContainer
XcpDaqList : EcucParamConfContainerDef
lowerMultiplicity = 1
upperMultiplicity = *
+destination
+subContainer
XcpEventChannel : EcucParamConfContainerDef
lowerMultiplicity = 1
upperMultiplicity = *
+reference
XcpEventChannelTriggeredDaqListRef : EcucReferenceDef
lowerMultiplicity = 1
upperMultiplicity = *
+parameter
XcpEventChannelNumber : EcucIntegerParamDef
min = 0
max = 65534
symbolicNameValue = true
+parameter
XcpEventChannelMaxDaqList : EcucIntegerParamDef
min = 0
max = 255
+parameter
XcpEventChannelPriority : EcucIntegerParamDef
min = 0
max = 255
+parameter
XcpEventChannelConsistency : EcucEnumerationParamDef
lowerMultiplicity = 1
upperMultiplicity = 1
defaultValue = ODT
+literal
ODT : EcucEnumerationLiteralDef
+literal
DAQ : EcucEnumerationLiteralDef
+literal
EVENT : EcucEnumerationLiteralDef
+parameter
XcpEventChannelType : EcucEnumerationParamDef
lowerMultiplicity = 1
upperMultiplicity = 1
defaultValue = DAQ
+literal
DAQ : EcucEnumerationLiteralDef
+literal
STIM : EcucEnumerationLiteralDef
+literal
DAQ_STIM : EcucEnumerationLiteralDef
+parameter
XcpEventChannelTimeCycle : EcucIntegerParamDef
min = 0
max = 255
lowerMultiplicity = 1
upperMultiplicity = 1
+parameter
XcpEventChannelTimeUnit : EcucEnumerationParamDef
lowerMultiplicity = 0
upperMultiplicity = 1
+literal
TIMESTAMP_UNIT_1NS : EcucEnumerationLiteralDef
+literal
TIMESTAMP_UNIT_10NS : EcucEnumerationLiteralDef
+literal
TIMESTAMP_UNIT_100NS : EcucEnumerationLiteralDef
+literal
TIMESTAMP_UNIT_1US : EcucEnumerationLiteralDef
+literal
TIMESTAMP_UNIT_10US : EcucEnumerationLiteralDef
+literal
TIMESTAMP_UNIT_100US : EcucEnumerationLiteralDef
+literal
TIMESTAMP_UNIT_1MS : EcucEnumerationLiteralDef
+literal
TIMESTAMP_UNIT_10MS : EcucEnumerationLiteralDef
+literal
TIMESTAMP_UNIT_100MS : EcucEnumerationLiteralDef
+literal
TIMESTAMP_UNIT_1PS : EcucEnumerationLiteralDef
+literal
TIMESTAMP_UNIT_10PS : EcucEnumerationLiteralDef
+literal
TIMESTAMP_UNIT_100PS : EcucEnumerationLiteralDef
+literal
TIMESTAMP_UNIT_1S : EcucEnumerationLiteralDef

Figure 12: Diagram XcpOdtEntry

10.2.9 XcpEventChannel

SWS Item	**ECUC_Xcp_00150 :**
Container Name	XcpEventChannel{XCP_EVENT_CHANNEL}
Description	This container contains the configuration of event channels on the XCP slave.
Configuration Parameters	

SWS Item	**ECUC_Xcp_00171 :**		
Name	XcpEventChannelConsistency {XCP_EVENT_CHANNEL_CONSISTENCY}		
Description	Type of consistency used by event channel		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	DAQ	Consistency on DAQ list level	
	EVENT	Consistency on Event Channel Level	
	ODT	Consistency on ODT level (default value). (default)	
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Xcp_00153 :**		
Name	XcpEventChannelMaxDaqList {XCP_MAX_DAQ_LIST}		
Description	Maximum amount of DAQ lists that are handled by this event channel.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_Xcp_00152 :**		
Name	XcpEventChannelNumber {XCP_EVENT_CHANNEL_NUMBER}		
Description	Index number of the event channel.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 65534		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_Xcp_00154 :**
Name	XcpEventChannelPriority
Description	Priority of the event channel
Multiplicity	1
Type	EcucIntegerParamDef

Range	0 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_Xcp_00173 :**		
Name	XcpEventChannelTimeCycle {XCP_EVENT_CHANNEL_TIME_CYCLE}		
Description	The event channel time cycle indicates which sampling period is used to process this event channel. A value of 0 means 'Not cyclic'.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Xcp_00174 :**		
Name	XcpEventChannelTimeUnit {XCP_EVENT_CHANNEL_TIME_UNIT}		
Description	This configuration parameter indicates the unit of the event channel time cycle.		
Multiplicity	0..1		
Type	EcucEnumerationParamDef		
Range	TIMESTAMP_UNIT_100MS	Unit is 100 millisecond.	
	TIMESTAMP_UNIT_100NS	Unit is 100 nanosecond.	
	TIMESTAMP_UNIT_100PS	Unit is 100 picosecond.	
	TIMESTAMP_UNIT_100US	Unit is 100 microsecond.	
	TIMESTAMP_UNIT_10MS	Unit is 10 millisecond.	
	TIMESTAMP_UNIT_10NS	Unit is 10 nanosecond.	
	TIMESTAMP_UNIT_10PS	Unit is 10 picosecond.	
	TIMESTAMP_UNIT_10US	Unit is 10 microsecond.	
	TIMESTAMP_UNIT_1MS	Unit is 1 millisecond.	
	TIMESTAMP_UNIT_1NS	Unit is 1 nonasecond.	
	TIMESTAMP_UNIT_1PS	Unit is 1 picosecond.	
	TIMESTAMP_UNIT_1S	Unit is 1 second.	
	TIMESTAMP_UNIT_1US	Unit is 1 microsecond.	
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local dependency: Dependent on the Parameter EventChannelTimeCycle. When this parameter is set to 0, the entire event channel time unit parameter shall be ignored.		

SWS Item	**ECUC_Xcp_00172 :**	
Name	XcpEventChannelType {XCP_EVENT_CHANNEL_TYPE}	
Description	This configuration parameter indicates what kind of DAQ list can be allocated to this event channel.	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	DAQ	only DAQ supported (default value). (default)
	DAQ_STIM	Both DAQ and STIM supported (Simultaneously).

	STIM	only STIM supported	
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Xcp_00151 :**		
Name	XcpEventChannelTriggeredDaqListRef		
Description	References all DAQ lists that are trigged by this event channel.		
Multiplicity	1..*		
Type	Reference to [XcpDaqList]		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

No Included Containers

10.2.10 XcpPdu

SWS Item	**ECUC_Xcp_00100 :**
Choice container Name	XcpPdu{XCP_PDU}
Description	Contains PDU information. A PDU may be either a transmission PDU or a reception PDU.

Container Choices		
Container Name	***Multiplicity***	***Scope / Dependency***
XcpRxPdu	0..1	This container specifies received PDUs.
XcpTxPdu	0..1	This container specifies transmission PDUs.

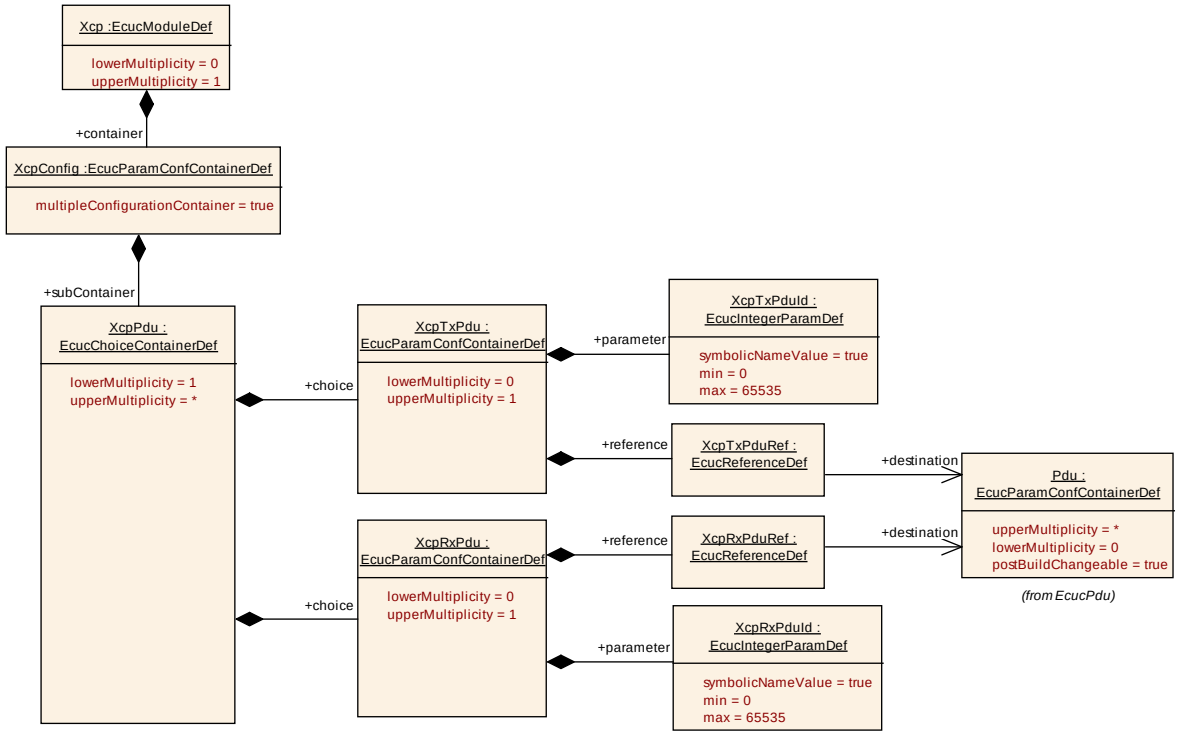

Figure 13: Diagram XcpPdu

10.2.11 XcpRxPdu

SWS Item	**ECUC_Xcp_00105 :**
Container Name	XcpRxPdu{XCP_RX_PDU}
Description	This container specifies received PDUs.
Configuration Parameters	

SWS Item	**ECUC_Xcp_00106 :**		
Name	XcpRxPduId {XCP_RX_PDU_ID}		
Description	ID of the PDU that will be received via a Xcp_<module>RxIndication.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 65535		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_Xcp_00107 :**		
Name	XcpRxPduRef {XCP_PDU_REF}		
Description	--		
Multiplicity	1		
Type	Reference to [Pdu]		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

No Included Containers

10.2.12 XcpTxPdu

SWS Item	ECUC_Xcp_00101 :		
Container Name	XcpTxPdu{XCP_TX_PDU}		
Description	This container specifies transmission PDUs.		
Configuration Parameters			

SWS Item	ECUC_Xcp_00103 :		
Name	XcpTxPduId {XCP_TX_PDU_ID}		
Description	The PDU identifier, which has to be used by the lower layer BSW module for TxConfirmations or TriggerTransmits.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 65535		
Default value	--		
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	ECUC_Xcp_00104 :		
Name	XcpTxPduRef {XCP_PDU_REF}		
Description	Reference to the external PDU definition.		
Multiplicity	1		
Type	Reference to [Pdu]		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

No Included Containers

10.3 Published Information

For details refer to the chapter 10.3 “Published Information” in *SWS_BSWGeneral.*

11 Not applicable requirements

[SWS_Xcp_00999]「These requirements are not applicable to this specification.」
(SRS_BSW_00171, SRS_BSW_00170, SRS_BSW_00387, SRS_BSW_00375, SRS_BSW_00416, SRS_BSW_00168, SRS_BSW_00423, SRS_BSW_00425, SRS_BSW_00426, SRS_BSW_00427, SRS_BSW_00428, BSW00431, SRS_BSW_00432, BSW00434, SRS_BSW_00336, SRS_BSW_00417, SRS_BSW_00161, SRS_BSW_00162, SRS_BSW_00005, SRS_BSW_00415, SRS_BSW_00164, SRS_BSW_00325, SRS_BSW_00326, SRS_BSW_00413, SRS_BSW_00347, SRS_BSW_00335, SRS_BSW_00410, SRS_BSW_00370, SRS_BSW_00314, SRS_BSW_00328, SRS_BSW_00312, SRS_BSW_00006, SRS_BSW_00377, SRS_BSW_00306, SRS_BSW_00309, SRS_BSW_00371, SRS_BSW_00360, SRS_BSW_00329, SRS_BSW_00330, SRS_BSW_00331, SRS_BSW_00009, SRS_BSW_00401, SRS_BSW_00172, SRS_BSW_00010, SRS_BSW_00333, SRS_BSW_00321, SRS_BSW_00341, SRS_Xcp_29008)