Document Title	Specification of Module XCP
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	412
Document Classification	Standard

Document Version	2.0.0
Document Status	Final
Part of Release	4.0
Revision	3

Document Change History

Date	Version	Changed by	Change Description
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 I**nter**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

3.1.1 Related standards and norms

[12] ASAM XCP – The Universal Measurement and Calibration Protocol
http://www.asam.net/index.php?option=com_content&task=view&id=136&Itemid=18
[13] ASAM XCP – Transport Layer Specification XCP on CAN
http://www.asam.net/index.php?option=com_content&task=view&id=136&Itemid=18
[14] ASAM XCP – Transport Layer Specification XCP on Ethernet
http://www.asam.net/index.php?option=com_content&task=view&id=136&Itemid=18
[15] ASAM XCP – Transport Layer Specification XCP on FlexRay
http://www.asam.net/index.php?option=com_content&task=view&id=136&Itemid=18

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 [12]

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

[**Xcp501**]
「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.」(BSW00380, BSW00419, BSW00383, BSW00346, BSW158)

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

[**Xcp500**]
「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

[**Xcp502**]「

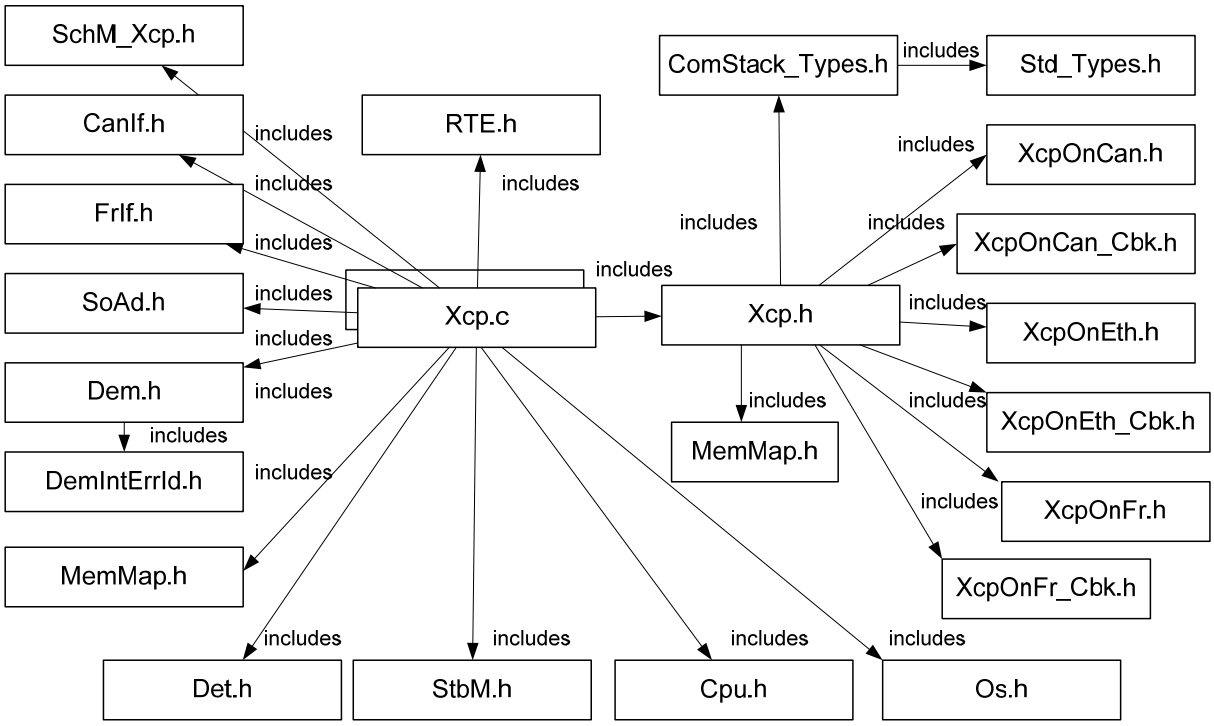


Figure 5-1: XCP Header File Structure」(BSW00381, BSW00412, BSW00409, BSW00301)

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*.

[**Xcp503**]「
All files related to the XCP module shall follow the naming convention *Xcp[_<description>].<extension>*」(BSW00300)

[**Xcp505**]
「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.」(BSW00302)

[**Xcp506**]
「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.」()

[**Xcp507**]「
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.」()

[**Xcp508**]「
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.」()

[**Xcp509**]「

The 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.

6 Requirements traceability

6.1 General Requirements on Basic Software Modules

Requirement		*Satisfied by*
BSW00344	Reference to link-time configuration	Xcp741
BSW00404	Reference to post build time configuration	Xcp742
BSW00405	Reference to multiple configuration sets	Xcp803
BSW00345	Pre-compile-time configuration	Xcp742
BSW159	Tool-based configuration	Xcp102
BSW167	Static configuration checking	Xcp103 Xcp104 Xcp105
BSW171	Configurability of optional functionality	n/a
BSW170	Data for reconfiguration of AUTOSAR SW-Components	n/a
BSW00380	Separate C-Files for configuration parameters	Xcp501
BSW00419	Separate C-Files for pre-compile time configuration parameters	Xcp501
BSW00381	Separate configuration header file for pre-compile time parameters	Xcp502
BSW00412	Separate H-File for configuration parameters	Xcp502
BSW00383	List dependencies of configuration files	Xcp501
BSW00384	List dependencies to other modules	Chapter 5
BSW00387	Specify the configuration class of callback function	n/a
BSW00388	Introduce containers	Xcp101
BSW00389	Containers shall have names	Chapter 10.2
BSW00390	Parameter content shall be unique within the module	Chapter 10.2
BSW00391	Parameter shall have unique names	Chapter 10.2
BSW00392	Parameters shall have a type	Chapter 10.2
BSW00393	Parameters shall have a range	Chapter 10.2
BSW00394	Specify the scope of the parameters	Chapter 10.2
BSW00395	List the required parameters (per parameter)	Chapter 10.2
BSW00396	Configuration classes	Chapter 10.2
BSW00397	Pre-compile-time parameters	Chapter 10.2
BSW00398	Link-time parameters	Chapter 10.2
BSW00399	Loadable Post-build time parameters	Chapter 10.2
BSW00400	Selectable Post-build time parameters	Chapter 10.2
BSW00402	Published information	Xcp807
BSW00375	Notification of wake-up reason	n/a
BSW101	Initialization interface	Xcp803
BSW00416	Sequence of Initialization	n/a
BSW00406	Check module initialization	Xcp811
BSW168	Diagnostic Interface of SW components	n/a
BSW00407	Function to read out published parameters	Xcp807
BSW00423	Usage of SW-C template to describe BSW modules with AUTOSAR Interfaces	n/a
BSW00424	BSW main processing function task allocation	Xcp823
BSW00425	Trigger conditions for schedulable objects	n/a
BSW00426	Exclusive areas in BSW modules	n/a
BSW00427	ISR description for BSW modules	n/a
BSW00428	Execution order dependencies of main processing functions	n/a
BSW00429	Restricted BSW OS functionality access	Chapter 5.6
BSW00431	The BSW Scheduler module implements task bodies	n/a
BSW00432	Modules should have separate main processing functions for read/receive and write/transmit data path	n/a
BSW00433	Calling of main processing functions	Xcp823
BSW00434	The Schedule Module shall provide an API for exclusive areas	n/a
BSW00336	Shutdown interface	n/a
BSW00337	Classification of errors	Error

		classification
BSW00338	Detection and Reporting of development errors	Chapter 7.6 Chapter 7.7
BSW00369	Do not return development error codes via API	Chapter
BSW00339	Reporting of production relevant error status	Chapter
BSW00417	Reporting of Error Events by Non-Basic Software	n/a
BSW00323	API parameter checking	Chapter 8.3
BSW004	Version check	Xcp749
BSW00409	Header files for production code error IDs	Xcp502
BSW00385	List possible error notifications	Chapter 7.5
BSW00386	Configuration for detecting an error	Xcp754
BSW161	Microcontroller abstraction	n/a
BSW162	ECU layout abstraction	n/a
BSW005	No hard coded horizontal interfaces within MCAL	n/a
BSW00415	User dependent include files	n/a
BSW164	Implementation of interrupt service routines	n/a
BSW00325	Runtime of interrupt service routines	n/a
BSW00326	Transition from ISRs to OS tasks	n/a
BSW00342	Usage of source code and object code	Chapter 10.2.1
BSW00343	Specification and configuration of time	Chapter 10.2
BSW160	Human-readable configuration data	Xcp744
BSW007	HIS MISRA C	Xcp745
BSW00300	Module naming convention	Xcp503
BSW00413	Accessing instances of BSW modules	n/a
BSW00347	Naming separation of different instances of BSW drivers	n/a
BSW00305	Self-defined data types naming convention	Chapter 8.2
BSW00307	Global variables naming convention	Xcp800
BSW00310	API naming convention	Xcp800
BSW00373	Main processing function naming convention	Xcp823
BSW00327	Error values naming convention	Xcp763
BSW00335	Status values naming convention	n/a
BSW00350	Development error detection keyword	Xcp753
BSW00408	Configuration parameter naming convention	Xcp800
BSW00410	Compiler switches shall have defined values	n/a
BSW00411	Get version info keyword	Xcp807 Xcp808 Xcp809 Xcp810
BSW00346	Basic set of module files	Xcp501
BSW158	Separation of configuration from implementation	Xcp501
BSW00314	Separation of interrupt frames and service routines	n/a
BSW00370	Separation of callback interface from API	n/a
BSW00348	Standard type header	Chapter 5.9
BSW00353	Platform specific type header	Chapter 5.9
BSW00361	Compiler specific language extension header	Chapter 5.9
BSW00301	Limit imported information	Xcp502
BSW00302	Limit exported information	Xcp505
BSW00328	Avoid duplication of code	n/a
BSW00312	Shared code shall be reentrant	n/a
BSW006	Platform independency	n/a
BSW00357	Standard API return type	Chapter 8.3
BSW00377	Module specific API return types	n/a
BSW00304	AUTOSAR integer data types	Chapter 8.3
BSW00355	Do not redefine AUTOSAR integer data types	Chapter 8.3
BSW00378	AUTOSAR boolean type	Chapter 8.3
BSW00306	Avoid direct use of compiler and platform specific keywords	n/a
BSW00308	Definition of global data	Xcp760
BSW00309	Global data with read-only constraint	n/a

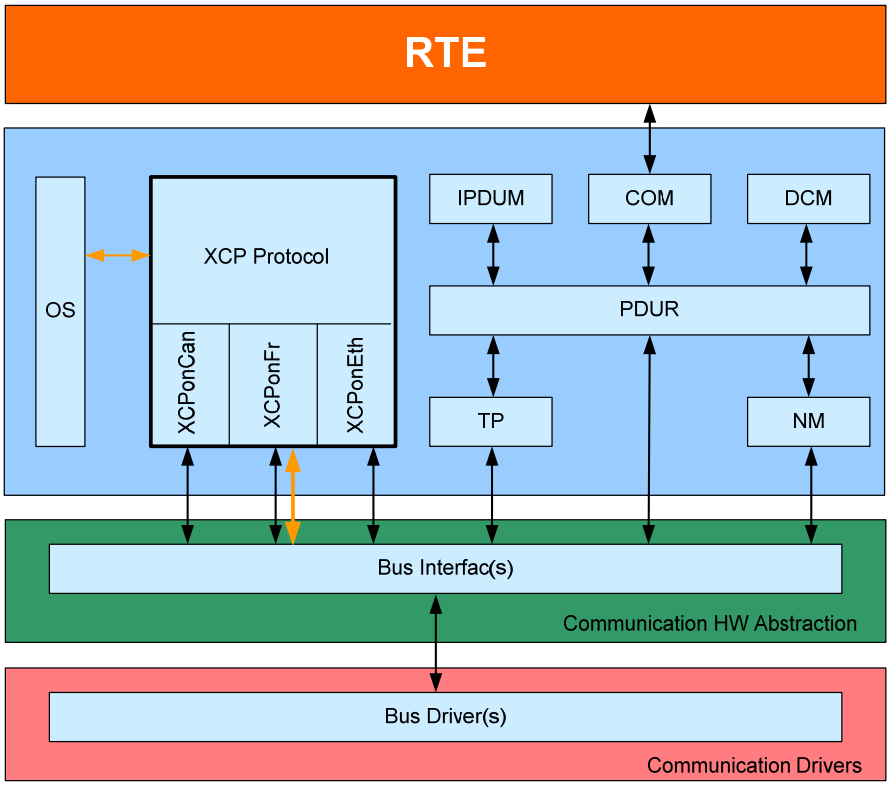
BSW00371	Do not pass function pointers via API	n/a
BSW00358	Return type of init() functions	Xcp803
BSW00414	Parameter of init function	Xcp803
BSW00376	Return type and parameters of main processing functions	Chapter 8.5
BSW00359	Return type of callback functions	Chapter 8.4
BSW00360	Parameters of callback functions	n/a
BSW00329	Avoidance of generic interfaces	n/a
BSW00330	Usage of macros / inline functions instead of functions	n/a
BSW00331	Separation of error and status values	n/a
BSW009	Module User Documentation	n/a
BSW00401	Documentation of multiple instances of configuration parameters	n/a
BSW172	Compatibility and documentation of scheduling strategy	n/a
BSW010	Memory resource documentation	n/a
BSW00333	Documentation of callback function context	n/a
BSW00374	Module vendor identification	Xcp807
BSW00379	Module identification	Xcp807
BSW003	Version identification	Xcp807
BSW00318	Format of module version numbers	Xcp807
BSW00321	Enumeration of module version numbers	n/a
BSW00341	Microcontroller compatibility documentation	n/a
BSW00334	Provision of XML file	Xcp751
BSW00435	Header File Structure for the Basic Software Scheduler	Xcp747
BSW00436	Module Header File Structure for the Memory Mapping	Xcp748

6.2 Requirements on the XCP Basic Software Module

Requirement		***Satisfied by***
BSW429001	Location of XCP within the architecture	Xcp701
BSW429002	API usage	Xcp712, Xcp714, Xcp720, Xcp734
BSW429003	Unique PDU-ID	Xcp702
BSW429004	XCP Specification Version 1.1	Xcp703
BSW429005	XCP on CAN	Xcp713
BSW429006	XCP on FlexRay	Xcp719
BSW429007	XCP on Ethernet	Xcp733
BSW429008	A2L Support	n/a
BSW429009	Synchronous data acquisition	Xcp705
BSW429010	Synchronous data stimulation	Xcp707
BSW429011	Block communication mode	Xcp711
BSW429012	Interleaved communication mode	Xcp710
BSW429013	Dynamic data transfer configuration	Xcp706
BSW429014	Timestamped Data transfer	Xcp709
BSW429015	Bypassing	Xcp761
BSW429016	Seed & Key	Xcp766
BSW429017	XCP Initialization	Xcp803

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.



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

[**Xcp701**]⌈
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.⌋(BSW429001)

[**Xcp702**]⌈
For transmitting and receiving of XCP messages, unique PDU-IDs shall be used. ⌋(BSW429003)

[**Xcp703**]⌈
The AUTOSAR XCP Module shall support the ASAM XCP Specification Version 1.1. ⌋(BSW429004)

[**Xcp705**]⌈

The AUTOSAR XCP Module shall support the basic feature “Synchronous data acquisition (measurement) “. Please refer to [13] ⌋(BSW429009)

[**Xcp706**]⌈
The AUTOSAR XCP Module shall support the feature “Dynamic DAQ Configuration”. according to [13] ⌋(BSW429013)

[**Xcp707**]
⌈The AUTOSAR XCP Module shall support the basic feature “Synchronous data stimulation” according to [13]⌋(BSW429010)

[**Xcp708**]⌈
The AUTOSAR XCP Module shall support the basic feature “Online memory calibration (read / write access) “, according to [13]⌋()

[**Xcp709**]⌈
The AUTOSAR XCP Module shall support the feature “Timestamped Data Transfer“, according to [13]⌋(BSW429014)

[**XCP 768**]⌈
The ECU local time shall be derived from the AUTOSAR OS.⌋()

[**Xcp711**]⌈
The AUTOSAR XCP Module shall support the feature “Block communication mode“, according to [13]⌋(BSW42911)

[**Xcp761**]⌈
The AUTOSAR XCP Module shall support the feature “Bypassing“, according to [13]⌋(BSW429015)

[**Xcp766**]⌈
The AUTOSAR XCP Module shall support the feature “Seed & Key” according to [13]⌋(BSW429016)

[**Xcp712**]⌈
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.⌋(BSW429002)

7.1 XCP on CAN

[**Xcp713**]⌈
The AUTOSAR XCP Module shall support the CAN communications bus according to [14]⌋(BSW429005)

[**Xcp714**]

⌜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 [9]⌟(BSW429002)

[**Xcp715**]⌜
For sending and receiving XCP data via CAN, at least two different CAN identifiers have to be configured to be used by XCP.⌟()

[**Xcp716**]⌜
Performance information shall be exchanged between the XCP master and XCP slave using the parameters according to [14]⌟()

[**Xcp718**]⌜
The XCP Module shall support the GET_SLAVE_ID command according to [14]⌟()

7.2 XCP on FlexRay

[**Xcp719**]⌜
The AUTOSAR XCP Module shall support the FlexRay communications bus according to [16]⌟(BSW429006)

[**Xcp720**]⌜
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 [8].⌟(BSW429002)

[**Xcp721**]⌜
All XCP on FlexRay LPDUs always are event driven. Please refer to Chapter 1.1.2 “FlexRay Frame Type” of [16]⌟()

[**Xcp722**]⌜
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.

[**Xcp723**]⌜
The usage of FlexRay Communication Controller’s hardware buffers shall be configured by the corresponding parameters according to [16]⌟()

[**Xcp724**]⌈
The FlexRay PDU length used by the AUTOSAR XCP module shall be set using the corresponding parameters according to [16]⌋()

[**Xcp725**]⌈
LPDU_IDs which shall be routed to the AUTOSAR XCP module (using the AUTOSAR Bus Interface) have to be defined by the system designer. ⌋()

[**Xcp726**]⌈
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. ⌋()

[**Xcp728**]⌈
The XCP master gets the information about the XCP dedicated FlexRay Communication Controller buffers from the ASAM MCD 2MC description file. ⌋()

[**Xcp729**]⌈
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].⌋()

[**Xcp730**]⌈
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]⌋()

[**Xcp731**]
⌈For XCP on FlexRay, the Tail consists of a Control Field containing optional FILL bytes according to [16].⌋()

[**Xcp732**]⌈
The AUTOSAR XCP module shall be able to pack multiple XCP messages into one FlexRay Frame according to [16].⌋()

7.3 XCP on Ethernet

[**Xcp733**]⌈
The AUTOSAR XCP Module shall support the Ethernet communications bus according to [15] ⌋(BSW429007)

[**Xcp734**]⌈
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 [10].⌋(BSW429002)

[**Xcp735**]⌈
The AUTOSAR XCP slave connected by Ethernet and TCP/IP or UDP/IP is addressed by its IP Address and Port number.⌋()

[**Xcp736**]⌈
The AUTOSAR XCP slave only accepts one connection at the time. ⌋()

[**Xcp737**]⌈
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.⌋()

[**Xcp738**]⌈
The addressing scheme is defined according to [15]⌋()
.

[**Xcp739**]⌈
The header and tail of an XCP on Ethernet message have to be set according to [15] ⌋()

[**Xcp740**]⌈
The upper performance limit depends on the protocol stack of the host system. The corresponding parameters defined according to [15] have to be set.⌋()

[**Xcp710**]⌈
The AUTOSAR XCP Module shall support the feature “Interleaved communication mode“, according to according to [15] ⌋(BSW429012)

7.4 Requirements on Debugging

[**Xcp760**]⌈
Each variable that shall be accessible by AUTOSAR Debugging, shall be defined as global variable.⌋(BSW00308)

[**Xcp759**]
⌈All type definitions of variables which shall be debugged, shall be accessible by the header file Xcp.h⌋()

[**Xcp762**]⌈
The declaration of variables in the header file shall be such that it is possible to calculate the size of the variables by C-"sizeof".⌋()

[**Xcp765**]⌈
Variables available for debugging shall be described in the respective Basic Software Module Description.⌋()

[**Xcp764**]⌈
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

[**Xcp741**]⌈
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⌋(BSW00344)

[**Xcp742**]
⌈The XCP module shall support pre-compile time, link-time and post-build-time configuration.⌋(BSW00404, BSW00345)

[**Xcp744**]⌈
The description of the configuration and initialization data itself is not part of this specification but very implementation specific. The generated configuration data should be “human-readable”.⌋(BSW160)

[**Xcp745**]⌈
The XCP module's implementation shall be conform to the HIS subset of the MISRA C Standard.⌋(BSW007)

[**Xcp746**]⌈
In case development error detection is enabled for the XCP module, the XCP module shall check API parameters for validity and report detected errors to the DET⌋()

[**Xcp747**]⌈
The XCP module source code file(s) shall include *SchM_Xcp.h* if data consistency mechanisms of the BSW scheduler are required as described in Code file structure⌋ (BSW00435)

[**Xcp748**]⌈
The XCP module header file shall include *MemMap.h* and apply the memory mapping abstraction mechanisms as specified by Code file structure⌋(BSW00436)

[**Xcp749**]⌈
The header file *Xcp.h* shall contain a software and specification version number.⌋ (BSW004)

[**Xcp751**]⌈
The XCP module shall provide an XML file that contains the data which is required for the SW identification (it shall contain the vendor identification, module ID and software version information), configuration and integration process. This file should describe vendor specific configuration parameters as well as it should contain recommended configuration parameter values.⌋(BSW00334)

7.5 Error classification

[**Xcp752**]
⌈Values for production code Event Ids are assigned externally by the configuration of the Dem. They are published in the file Dem_IntErrId.h and included via Dem.h.⌋()

[**Xcp763**]
⌈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.⌋(BSW00327)

[**Xcp753**]
⌈Development error values are of type uint8.

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	Production	XCP_E_INIT_FAILED	Assigned by DEM
Null pointer has been passed as an argument	Development	XCP_E_NULL_POINTER	0x12

7.6 Error detection

[**Xcp754**]⌈
The detection of development errors is configurable (*ON* / *OFF*) at pre-compile time. The switch XCP_*DEV_ERROR_DETECT* (see chapter 10) shall activate or deactivate the detection of all development errors.⌋(BSW00386)

[**Xcp755**]
⌈If the XCP_*DEV_ERROR_DETECT* switch is enabled, API parameter checking is enabled. The detailed description of the detected errors can be found in chapter 8.3.⌋
()

[**Xcp756**]
⌈The detection of production code errors cannot be switched off.⌋()

7.7 Error notification

[**Xcp757**]
⌈Detected development errors shall be reported to the *Det_ReportError* service of the Development Error Tracer (DET) if the pre-processor switch *XCP_DEV_ERROR_DETECT* is set (see chapter 10). ⌋()

[**Xcp758**] ⌈
Production errors shall be reported to Diagnostic Event Manager.⌋()

7.8 Version checking

[**XCP767**] ⌈The XCP module shall perform Inter Module Checks to avoid integration of incompatible files.⌋()
The imported included files shall be checked by preprocessing directives.

The following version numbers shall be verified:
- <MODULENAME>_AR_RELEASE_MAJOR_VERSION
- <MODULENAME>_AR_RELEASE_MINOR_VERSION
Where <MODULENAME> is the module abbreviation of the other (external) modules

which provide header files included by the XCP module.

If the values are not identical to the expected values, an error shall be reported.

8 API specification

[**Xcp800**]⌈
All functions or global variables, whether they are specified or not shall follow the naming scheme `Xcp_<name>`, where the first letter of each word in <name> is written uppercase and the remainder of the word lowercase.⌋(BSW00307, BSW00310, BSW00408)

8.1 Imported types

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

[**Xcp801**]⌈

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

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

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

[**Xcp803**] ⌈

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.	

⌋(BSW00405, BSW101, BSW00358, BSW00414, BSW429017)

[**Xcp802**] ⌈The function `Xcp_Init` shall internally store the configuration address to enable subsequent API calls to access the configuration⌋()

[**Xcp834**] ⌈
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

[**Xcp807**] ⌈

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.	

⌋(BSW00402, BSW00407, BSW00411, BSW00374, BSW00379, BSW003, BSW00318)

[**Xcp808**] ⌈

The function `Xcp_GetVersionInfo` shall return the version information of this module. The version information includes:

- Module Id
- Vendor Id
- Vendor specific version numbers (BSW00407).⌋(BSW00411)

[**Xcp809**] ⌈

The function `Xcp_GetVersionInfo` shall be pre compile time configurable On/Off by the configuration parameter: `XCP_VERSION_INFO_API`⌋(BSW00411)

[**Xcp810**] ⌈

If source code for caller and callee of `Xcp_GetVersionInfo` is available, the XCP should realize `Xcp_GetVersionInfo` as a macro, defined in the module's header file.⌋(BSW00411)

[**Xcp825**] ⌈

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

[**Xcp836**] ⌈
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

[**Xcp813**] ⌈

Service name:	Xcp_<module>RxIndication	

Syntax:	`void Xcp_<module>RxIndication(`
	`  PduIdType XcpRxPduId,`
	`  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:	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 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.

[**XCP844**] ⌈
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

[**Xcp814**] ⌈

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.

[**Xcp840**] ⌈
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 `E_NOT_OK`.⌋()

[**XCP841**] ⌈
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

[**Xcp835**] ⌈

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

		shall be copied to. 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 has been copied. PduInfoPtr must not be used since it may contain a NULL pointer or point to invalid data.
Description:	The lower layer communication module requests the buffer of the SDU for transmission from the upper layer module.	

」()

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.

[**Xcp842**] 「
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`.」()

[**XCP843**] 「
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

[**Xcp844**]

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.	

[**Xcp845**]The XCP module shall provide this service only if XCP_SUPPRESS_TX_SUPPORT (see XCP169_Conf) equals TRUE.

[**Xcp846**]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.

[**Xcp847**]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

[**Xcp823**] ⌈

Service name:	Xcp_MainFunction

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

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

⌋(BSW00424, BSW00433, BSW00373)

[**Xcp824**] ⌈

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 CanTxPduId and CAN related data in the L-PDU 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.
FrIf_Transmit	Requests the sending of a PDU.
SoAdIf_Transmit	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

[**Xcp832**] ⌈

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.

⌋()

[**Xcp833**] ⌈

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 Device Driver(s).

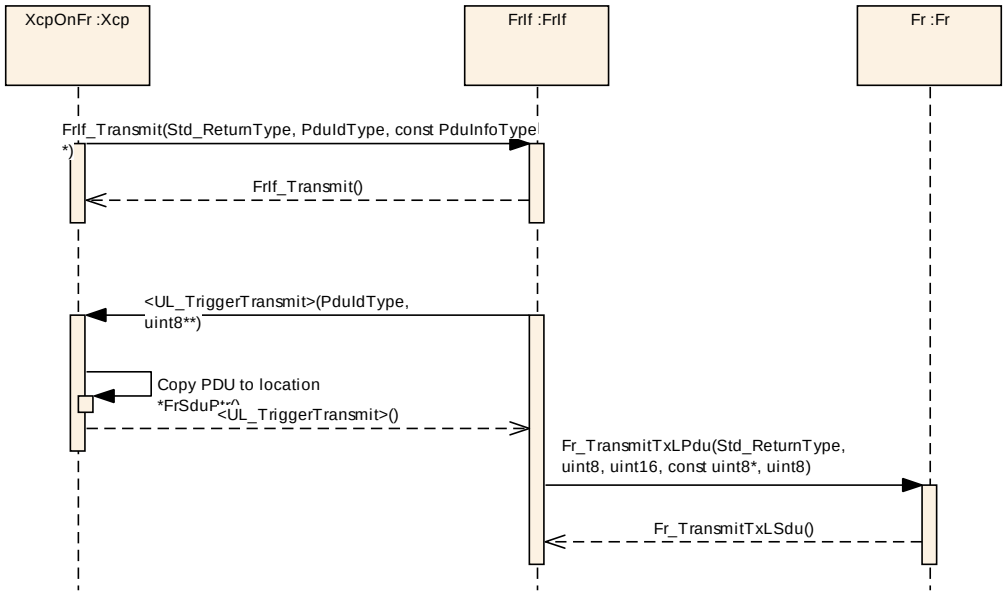
[**Xcp831**]「

API function	***Description***
<Cdd_Transmit>	Requests the sending of a PDU via Complex Device 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.
」()	

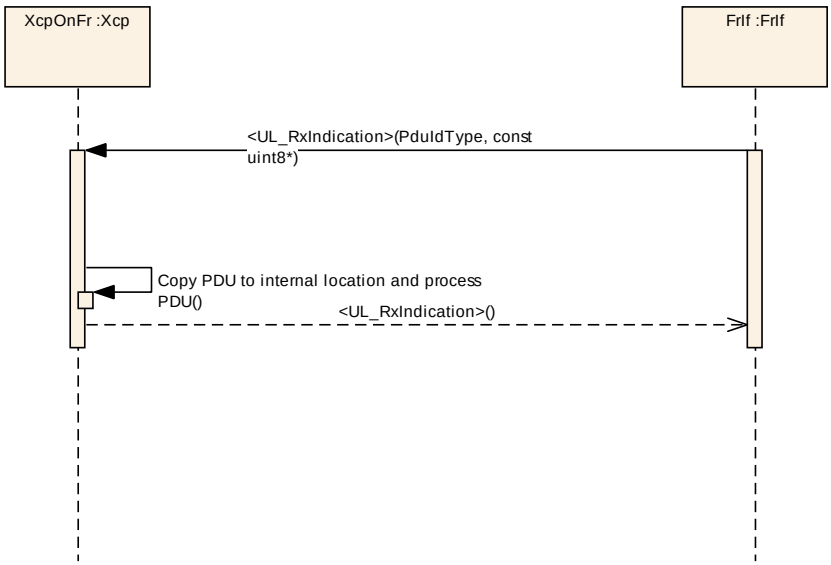
9 Sequence diagrams

9.1 XCP on FlexRay

9.1.1 Xcp on FlexRay Transmit

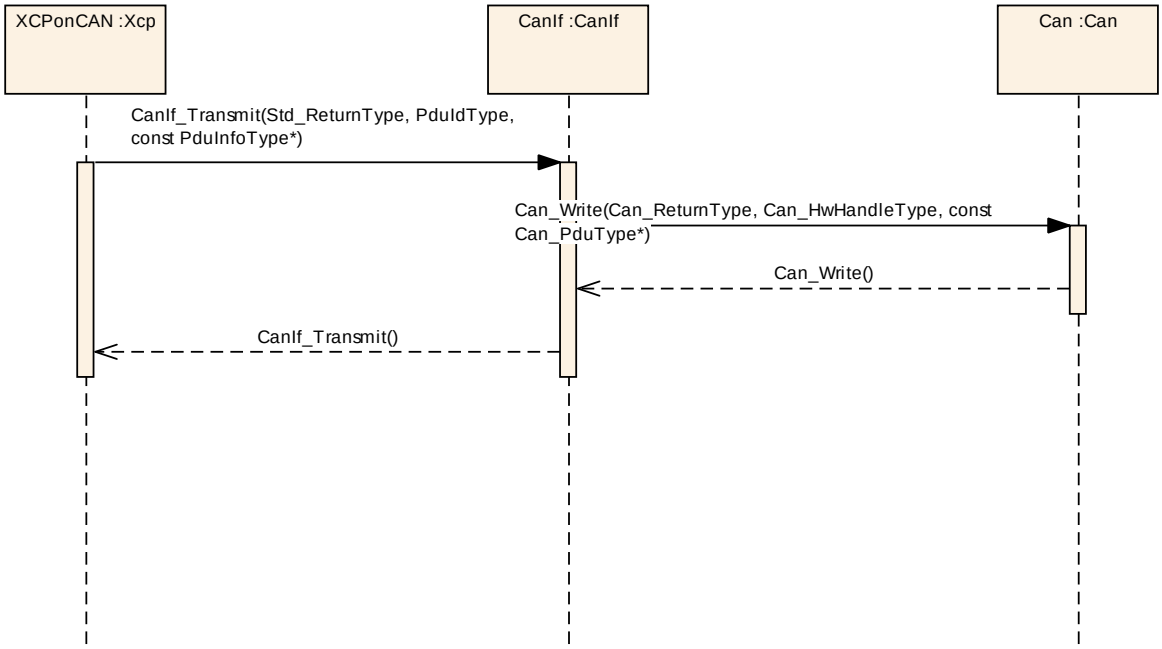


9.1.2 Xcp on FlexRay Receive Indication

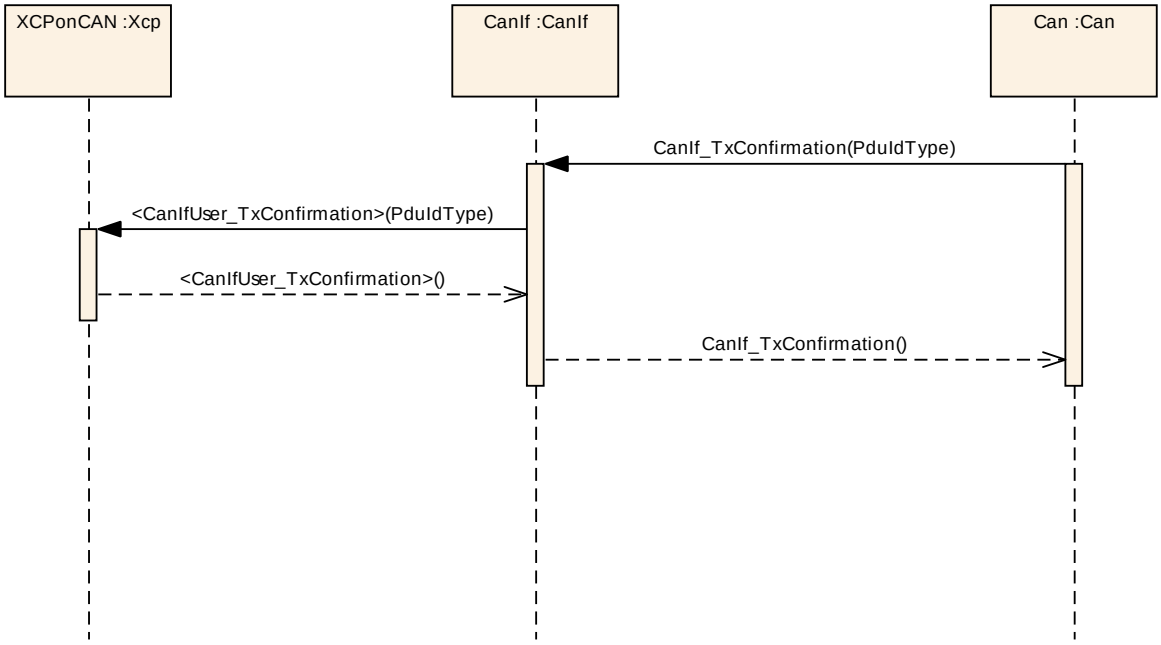


9.2 XCP on CAN

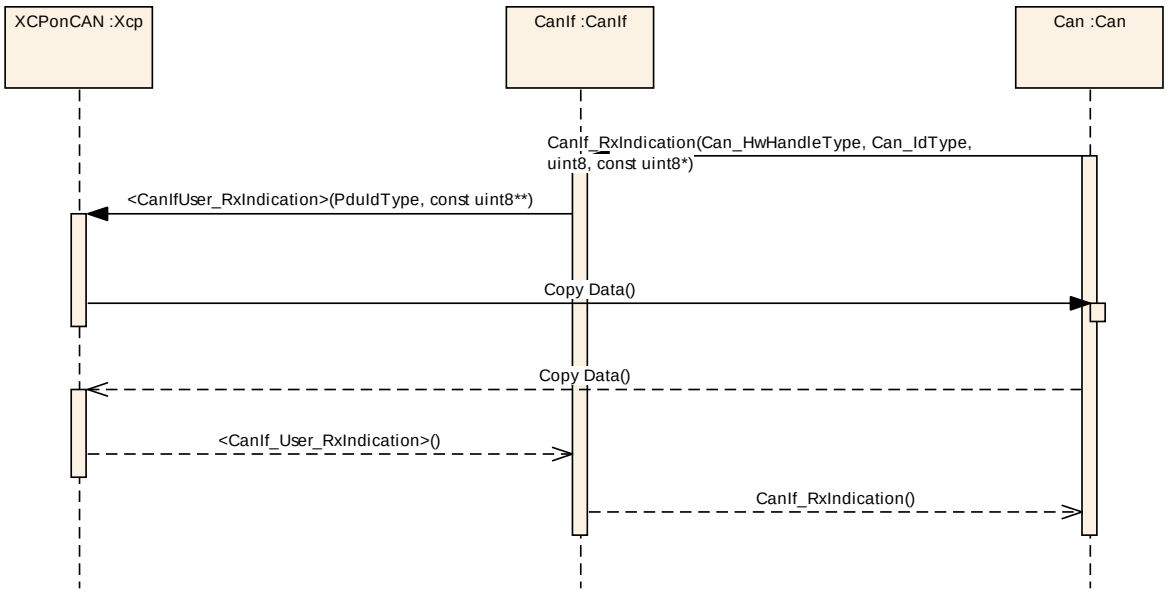
9.2.1 Xcp on CAN Transmit



9.2.2 Xcp on CAN Transmit Confirmation

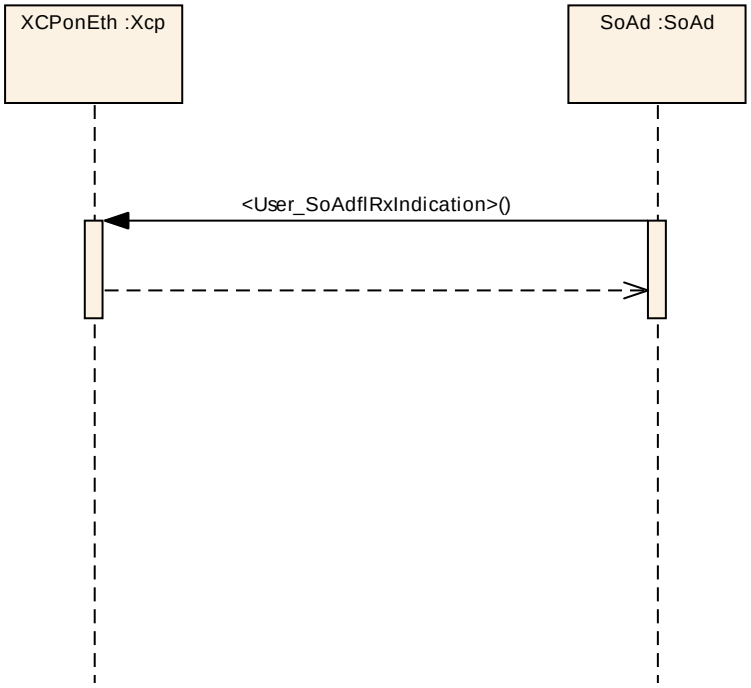


9.2.3 Xcp on CAN Receive Indication



9.3 XCP on Ethernet

9.3.1 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

In addition to this section, it is highly recommended to read the documents:

- AUTOSAR Layered Software Architecture
- AUTOSAR ECU Configuration Specification

The following is only a short survey of the topic and it will not replace the ECU Configuration Specification document.

10.1.1 Configuration and configuration parameters

Configuration parameters define the variability of the generic part(s) of an implementation of a module. This means that only generic or configurable module implementation can be adapted to the environment (software/hardware) in use during system and/or ECU configuration.

The configuration of parameters can be achieved at different times during the software process: before compile time, before link time or after build time. In the following, the term “configuration class” (of a parameter) shall be used in order to refer to a specific configuration point in time.

10.1.2 Variants

Variants describe sets of configuration parameters. E.g., variant 1: only pre-compile time configuration parameters; variant 2: mix of pre-compile- and post build time-configuration parameters. In one variant a parameter can only be of one configuration class.

10.1.3 Containers

[**Xcp101**] ⌈
Containers structure the set of configuration parameters. This means:
- *all* configuration parameters are kept in containers.
- (sub-) containers can reference (sub-) containers. It is possible to assign a multiplicity to these references. The multiplicity then defines the possible number of instances of the contained parameters.⌋(BSW00388)

10.1.4 Specification template for configuration parameters

The following tables consist of three sections:
- the general section
- the configuration parameter section
- the section of included/referenced containers

Pre-compile time - specifies whether the configuration parameter shall be of configuration class *Pre-compile time* or not

Label	***Description***
x	The configuration parameter shall be of configuration class *Pre-compile time*.
--	The configuration parameter shall never be of configuration class *Pre-compile time*.

Link time - specifies whether the configuration parameter shall be of configuration class *Link time* or not

Label	***Description***
x	The configuration parameter shall be of configuration class *Link time*.
--	The configuration parameter shall never be of configuration class *Link time*.

Post Build - specifies whether the configuration parameter shall be of configuration class *Post Build* or not

Label	***Description***
x	The configuration parameter shall be of configuration class *Post Build* and no specific implementation is required.
L	*Loadable* - the configuration parameter shall be of configuration class *Post Build* and only one configuration parameter set resides in the ECU.
M	*Multiple* - the configuration parameter shall be of configuration class *Post Build* and is selected out of a set of multiple parameters by passing a dedicated pointer to the init function of the module.
--	The configuration parameter shall never be of configuration class *Post Build*.

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.

[**Xcp102**] ⌈
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.⌋(BSW159)

[**XCP103**] ⌈
The configuration tool must check the consistency of the configuration at configuration time.⌋(BSW167)

[**Xcp104**] ⌈
Configuration rules and constraints for plausibility checks shall be performed during configuration time, wherever possible.⌋(BSW167)

[**Xcp105**] ⌈
These dependencies between FlexRay Interface and FlexRay Driver configuration must be provided at configuration time by the configuration tools.⌋(BSW167)

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.

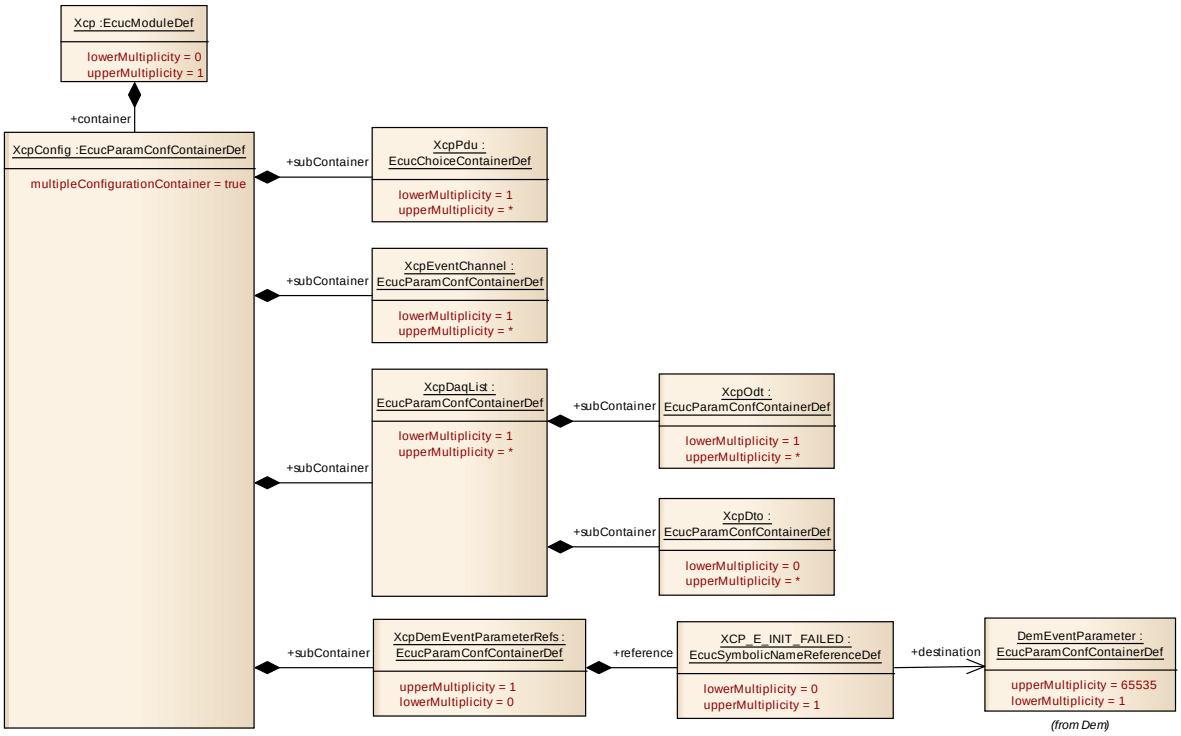
Use case: Execution time optimizations

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.



10.2.3 XcpGeneral

SWS Item	**XCP001_Conf :**
Container Name	XcpGeneral
Description	This container contains the general configuration parameters of the XCP.
Configuration Parameters	

SWS Item	**XCP164_Conf :**
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	**XCP012_Conf :**		
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	**XCP003_Conf :**		
Name	XcpDevErrorDetect {XCP_DEV_ERROR_DETECT}		
Description	Switches the Development Error Detection and Notification on or off. TRUE: Development Error Detection and Notificiation 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: Module		

SWS Item	**XCP170_Conf :**	
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: Module		

SWS Item	**XCP014_Conf :**		
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: Module		

SWS Item	**XCP004_Conf :**		
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: Module		

SWS Item	**XCP005_Conf :**		
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: Module		

SWS Item	**XCP011_Conf :**
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	**XCP013_Conf :**		
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	**XCP054_Conf :**		
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	**XCP059_Conf :**		
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	**XCP177_Conf :**
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	**XCP178_Conf :**		
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	**XCP006_Conf :**		
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: Module		

SWS Item	**XCP009_Conf :**		
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: Module		

SWS Item	**XCP008_Conf :**		
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: Module		

SWS Item	**XCP007_Conf :**

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: Module		

SWS Item	**XCP169_Conf :**		
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: Module		

SWS Item	**XCP176_Conf :**		
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: Module		

SWS Item	**XCP167_Conf :**		
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: Module		

SWS Item	**XCP166_Conf :**
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

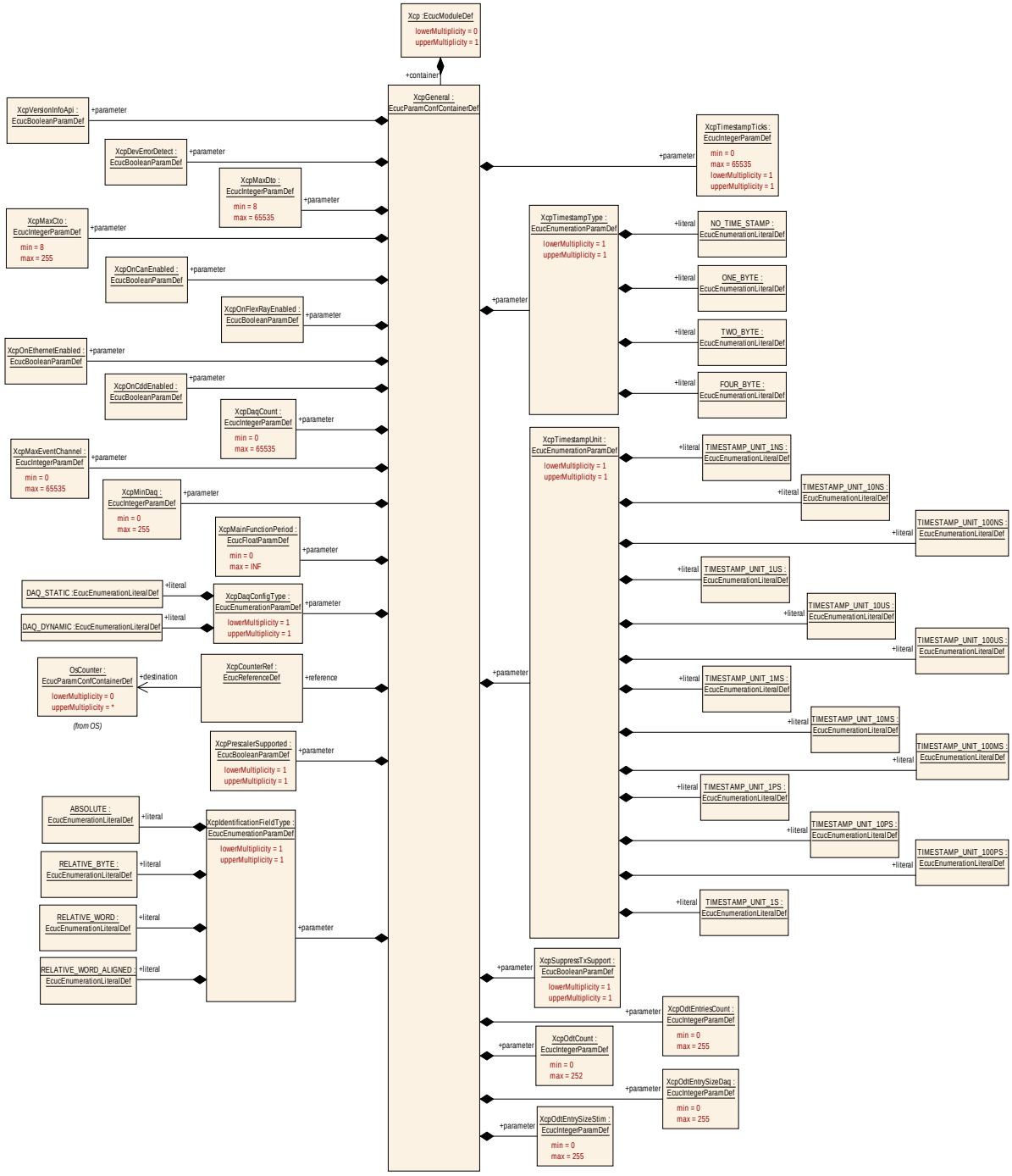
Range	FOUR_BYTE	timestamp field has the size of four byte.	
	NO_TIME_STAMP	timestamp field is not available.	
	ONE_BYTE	timestamp field has the size of one byte.	
	TWO_BYTE	timestamp field has the size of two byte.	
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**XCP168_Conf :**		
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: Module		

SWS Item	**XCP002_Conf :**		
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: Module		

SWS Item	**XCP162_Conf :**		
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			

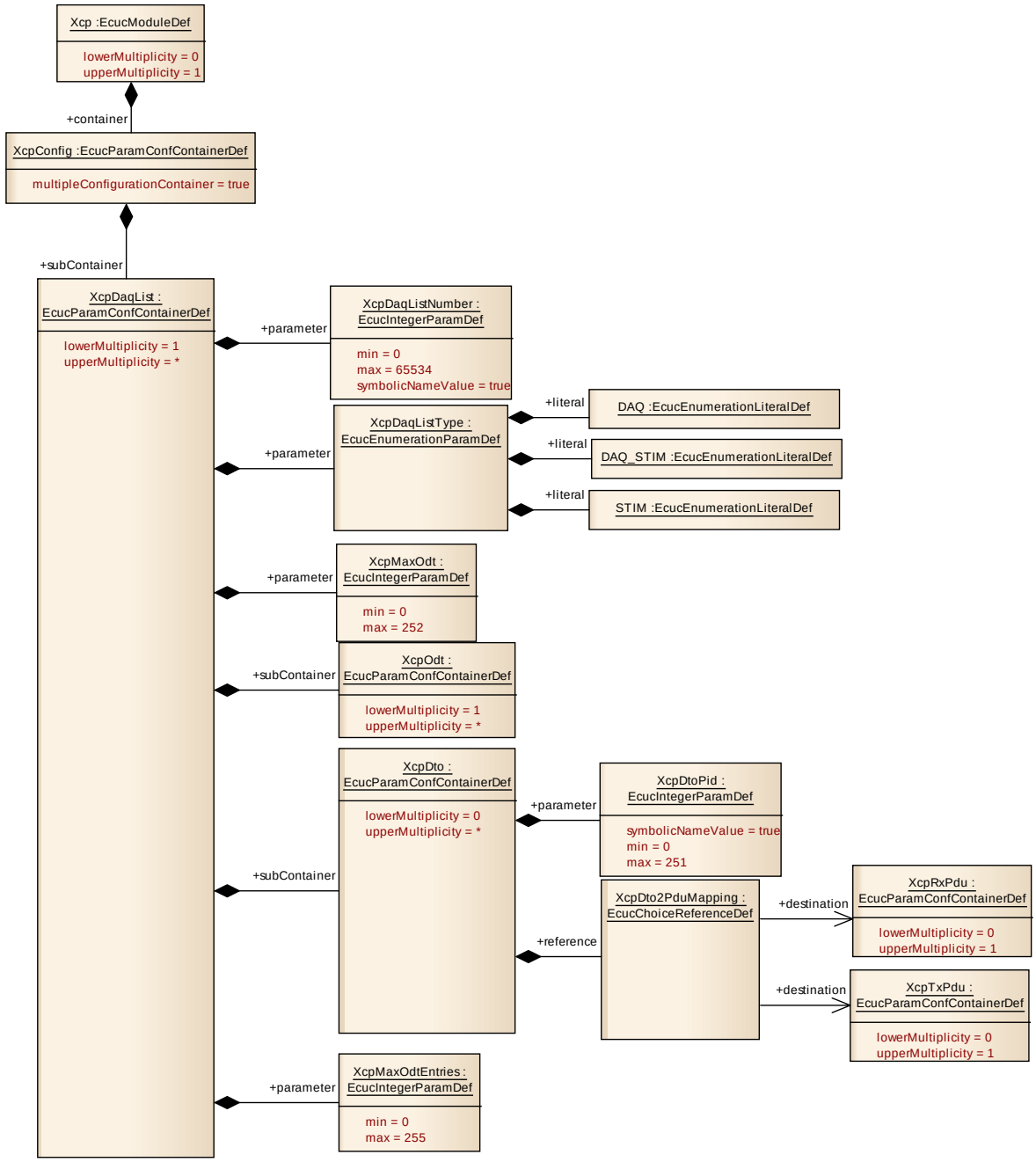
No Included Containers


Xcp :EcucModuleDef
lowerMultiplicity = 0
upperMultiplicity = 1
+container
XcpGeneral :
EcucParamConfContainerDef
XcpVersionInfoApi :
EcucBooleanParamDef
XcpDevErrorDetect :
EcucBooleanParamDef
XcpMaxDto :
EcucIntegerParamDef
min = 8
max = 65535
XcpMaxCto :
EcucIntegerParamDef
min = 8
max = 255
XcpOnCanEnabled :
EcucBooleanParamDef
XcpOnFlexRayEnabled :
EcucBooleanParamDef
XcpOnEthernetEnabled :
EcucBooleanParamDef
XcpOnCddEnabled :
EcucBooleanParamDef
XcpDaqCount :
EcucIntegerParamDef
min = 0
max = 65535
XcpMaxEventChannel :
EcucIntegerParamDef
min = 0
max = 65535
XcpMinDaq :
EcucIntegerParamDef
min = 0
max = 255
XcpMainFunctionPeriod :
EcucFloatParamDef
min = 0
max = INF
DAQ_STATIC :EcucEnumerationLiteralDef
DAQ_DYNAMIC :EcucEnumerationLiteralDef
XcpDaqConfigType :
EcucEnumerationParamDef
lowerMultiplicity = 1
upperMultiplicity = 1
OsCounter :
EcucParamConfContainerDef
lowerMultiplicity = 0
upperMultiplicity = *
(from OS)
XcpCounterRef :
EcucReferenceDef
XcpPrescalerSupported :
EcucBooleanParamDef
lowerMultiplicity = 1
upperMultiplicity = 1
ABSOLUTE :
EcucEnumerationLiteralDef
RELATIVE_BYTE :
EcucEnumerationLiteralDef
RELATIVE_WORD :
EcucEnumerationLiteralDef
RELATIVE_WORD_ALIGNED :
EcucEnumerationLiteralDef
XcpIdentificationFieldType :
EcucEnumerationParamDef
lowerMultiplicity = 1
upperMultiplicity = 1
XcpTimestampTicks :
EcucIntegerParamDef
min = 0
max = 65535
lowerMultiplicity = 1
upperMultiplicity = 1
XcpTimestampType :
EcucEnumerationParamDef
lowerMultiplicity = 1
upperMultiplicity = 1
NO_TIME_STAMP :
EcucEnumerationLiteralDef
ONE_BYTE :
EcucEnumerationLiteralDef
TWO_BYTE :
EcucEnumerationLiteralDef
FOUR_BYTE :
EcucEnumerationLiteralDef
XcpTimestampUnit :
EcucEnumerationParamDef
lowerMultiplicity = 1
upperMultiplicity = 1
TIMESTAMP_UNIT_1NS :
EcucEnumerationLiteralDef
TIMESTAMP_UNIT_10NS :
EcucEnumerationLiteralDef
TIMESTAMP_UNIT_100NS :
EcucEnumerationLiteralDef
TIMESTAMP_UNIT_1US :
EcucEnumerationLiteralDef
TIMESTAMP_UNIT_10US :
EcucEnumerationLiteralDef
TIMESTAMP_UNIT_100US :
EcucEnumerationLiteralDef
TIMESTAMP_UNIT_1MS :
EcucEnumerationLiteralDef
TIMESTAMP_UNIT_10MS :
EcucEnumerationLiteralDef
TIMESTAMP_UNIT_100MS :
EcucEnumerationLiteralDef
TIMESTAMP_UNIT_1PS :
EcucEnumerationLiteralDef
TIMESTAMP_UNIT_10PS :
EcucEnumerationLiteralDef
TIMESTAMP_UNIT_100PS :
EcucEnumerationLiteralDef
TIMESTAMP_UNIT_1S :
EcucEnumerationLiteralDef
XcpSuppressTxSupport :
EcucBooleanParamDef
lowerMultiplicity = 1
upperMultiplicity = 1
XcpOdtEntriesCount :
EcucIntegerParamDef
min = 0
max = 255
XcpOdtCount :
EcucIntegerParamDef
min = 0
max = 252
XcpOdtEntrySizeDaq :
EcucIntegerParamDef
min = 0
max = 255
XcpOdtEntrySizeStim :
EcucIntegerParamDef
min = 0
max = 255
+parameter
+literal
+destination
+reference

10.2.4 XcpConfig

SWS Item	**XCP020_Conf :**
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.
XcpDemEventParameterRefs	0..1	Container for the references to DemEventParameter elements which shall be invoked using the API Dem_ReportErrorStatus API in case the corresponding error occurs. The EventId is taken from the referenced DemEventParameter's DemEventId value. The standardized errors are provided in the container and can be extended by vendor specific error references.
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.



10.2.5 XcpDaqList

SWS Item	**XCP050_Conf :**
Container Name	XcpDaqList{XCP_DAQ_LIST}
Description	This container contains the configuration of the DAQs.
Configuration Parameters	

SWS Item	**XCP051_Conf :**
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	**XCP052_Conf :**		
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			

SWS Item	**XCP053_Conf :**		
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	**XCP058_Conf :**		
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	**XCP065_Conf :**
Container Name	XcpDto
Description	This container collects data transfer object specific parameters for the DAQ list.
Configuration Parameters	

SWS Item	**XCP066_Conf :**		
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			

SWS Item	**XCP067_Conf :**		
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			

No Included Containers

10.2.7 XcpOdt

SWS Item	**XCP055_Conf :**
Container Name	XcpOdt{XCP_ODT}
Description	This container contains ODT-specific parameter for the DAQ list.

Configuration Parameters

SWS Item	**XCP060_Conf :**		
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	**XCP057_Conf :**		
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	**XCP056_Conf :**		
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			

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	**XCP061_Conf :**
Container Name	XcpOdtEntry{XCP_ODT_ENTRY}
Description	This container collects all configuration parameters that comprise an ODT

	entry.
Configuration Parameters	

SWS Item	**XCP063_Conf :**		
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			

SWS Item	**XCP064_Conf :**		
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			

SWS Item	**XCP062_Conf :**		
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

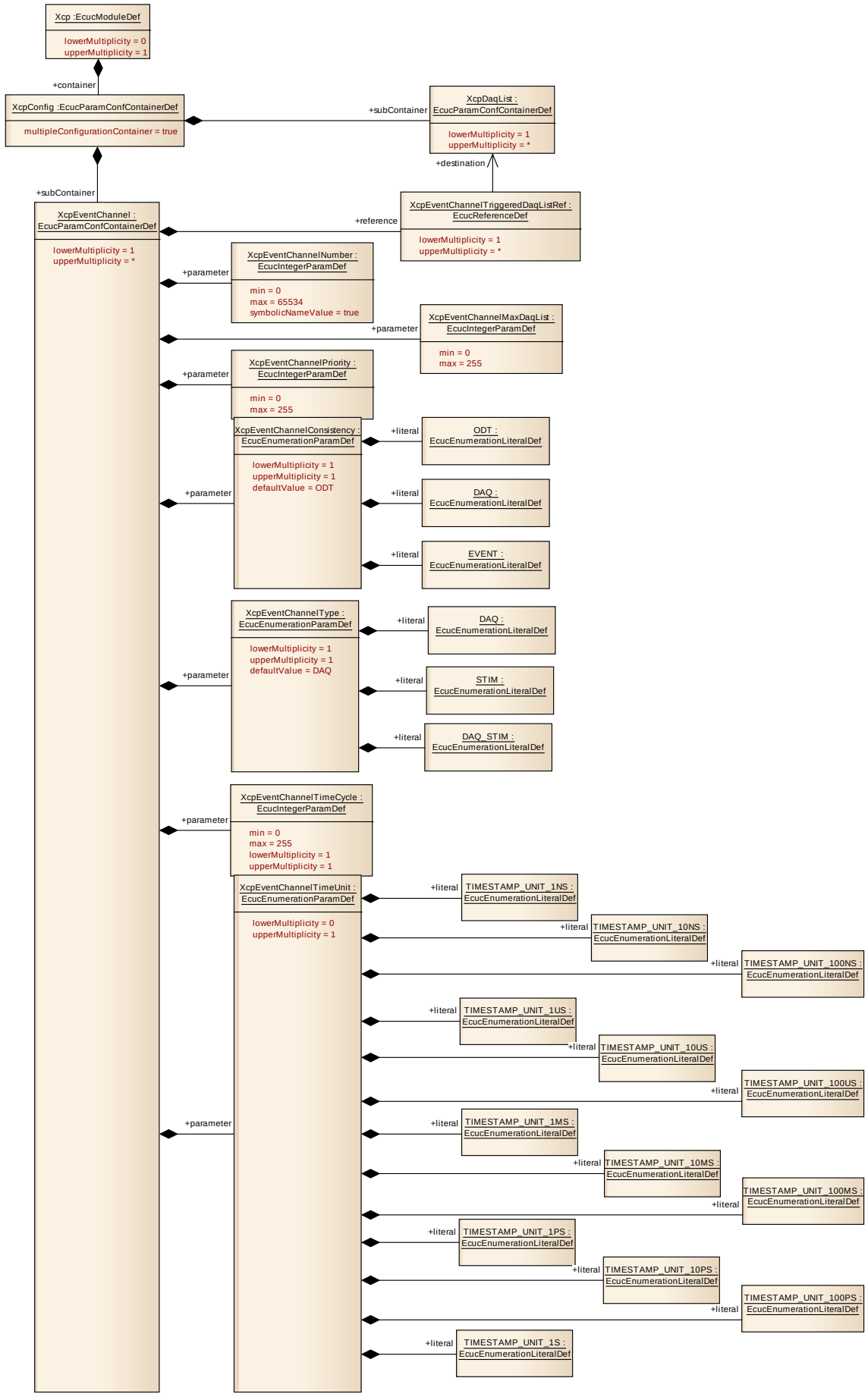
10.2.9 XcpDemEventParameterRefs

SWS Item	**Xcp160_Conf :**
Container Name	XcpDemEventParameterRefs
Description	Container for the references to DemEventParameter elements which shall be invoked using the API Dem_ReportErrorStatus API in case the

	corresponding error occurs. The EventId is taken from the referenced DemEventParameter's DemEventId value. The standardized errors are provided in the container and can be extended by vendor specific error references.
Configuration Parameters	

SWS Item	**Xcp161_Conf :**		
Name	XCP_E_INIT_FAILED		
Description	Reference to the DemEventParameter which shall be issued when the error "Initialization of XCP failed" has occured.		
Multiplicity	0..1		
Type	Reference to [DemEventParameter]		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency			

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

10.2.10 XcpEventChannel

SWS Item	**XCP150_Conf :**
Container Name	XcpEventChannel{XCP_EVENT_CHANNEL}
Description	This container contains the configuration of event channels on the XCP slave.
Configuration Parameters	

SWS Item	**XCP171_Conf :**		
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: Module		

SWS Item	**XCP153_Conf :**		
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			

SWS Item	**XCP152_Conf :**		
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			

SWS Item	**XCP154_Conf :**
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			

SWS Item	**XCP173_Conf :**		
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: Module		

SWS Item	**XCP174_Conf :**		
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: Module 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	**XCP172_Conf :**		
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: Module		

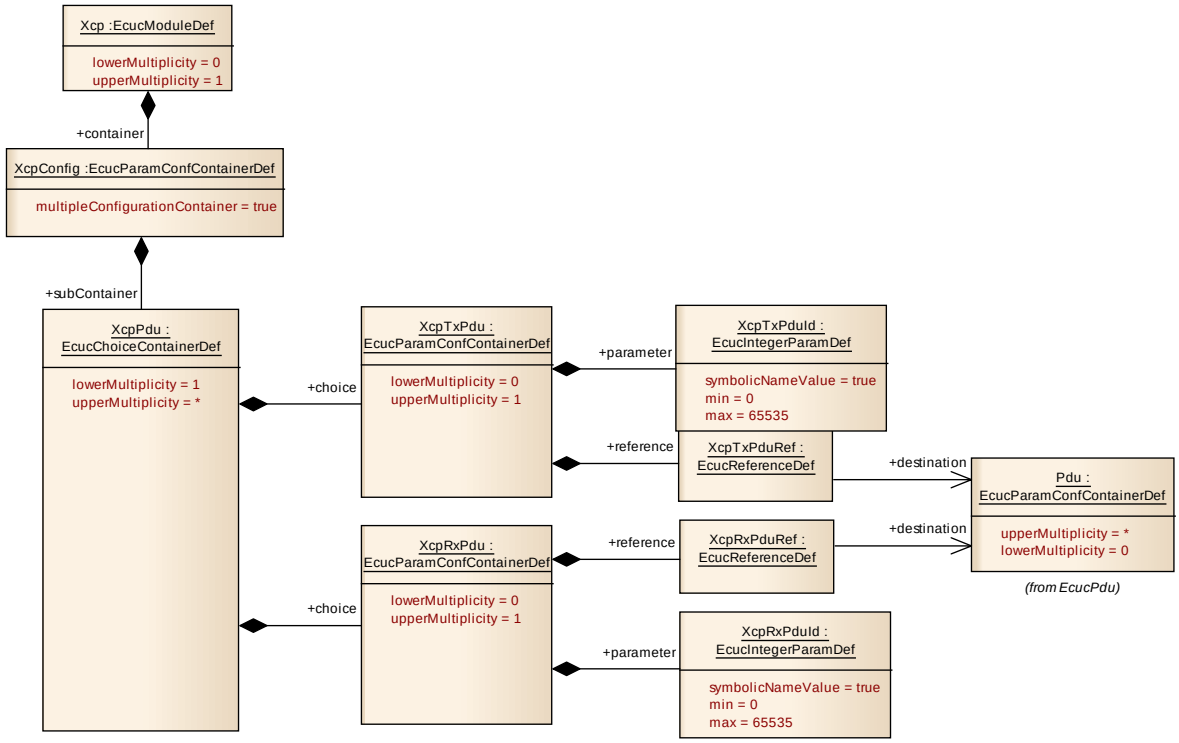
SWS Item	**XCP151_Conf :**		
Name	XcpEventChannelTriggeredDaqListRef		
Description	References all DAQ lists that are trigged by this event channel.		
Multiplicity	1..*		
Type	Reference to [XcpDaqList]		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

No Included Containers

10.2.11 XcpPdu

SWS Item	**XCP100_Conf :**
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.



10.2.12 XcpRxPdu

SWS Item	**XCP105_Conf :**
Container Name	XcpRxPdu{XCP_RX_PDU}
Description	This container specifies received PDUs.
Configuration Parameters	

SWS Item	**XCP106_Conf :**		
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			

SWS Item	**XCP107_Conf :**		
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

10.2.13 XcpTxPdu

SWS Item	XCP101_Conf :
Container Name	XcpTxPdu{XCP_TX_PDU}
Description	This container specifies transmission PDUs.
Configuration Parameters	

SWS Item	XCP103_Conf :		
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: Xcp and PduR/FrNm/FrTp		

SWS Item	XCP104_Conf :		
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

[Xcp843] 「The standardized common published parameters as required by BSW00402 in the SRS General on Basic Software Modules [3] shall be published within the header file of this module and need to be provided in the BSW Module Description. The according module abbreviation can be found in the List of Basic Software Modules [1].」()

Additional module-specific published parameters are listed below if applicable.

11 Not applicable requirements

[Xcp999] 「These requirements are not applicable to this specification.」(BSW171, BSW170, BSW00387, BSW00375, BSW00416, BSW168, BSW00423, BSW00425, BSW00426, BSW00427, BSW00428, BSW00431, BSW00432, BSW00434, BSW00336, BSW00417, BSW161, BSW162, BSW005, BSW00415, BSW164, BSW00325, BSW00326, BSW00413, BSW00347, BSW00335, BSW00410, BSW00370, BSW00314, BSW00328, BSW00312, BSW006, BSW00377, BSW00306, BSW00309, BSW00371, BSW00360, BSW00329, BSW00330, BSW00331, BSW009, BSW00401, BSW172, BSW010, BSW00333, BSW00321, BSW00341, BSW429008)