Elektrobit

EB tresos® AutoCore Generic 8 XCP

Stack documentation

product release 8.8.4

Elektrobit Automotive GmbH
Am Wolfsmantel 46
91058 Erlangen, Germany
Phone: +49 9131 7701 0
Fax: +49 9131 7701 6333
Email: info.automotive@elektrobit.com

Technical support

https://www.elektrobit.com/support

Legal disclaimer

Confidential information.

Table of Contents

1. Overview of EB tresos AutoCore Generic 8 XCP Stack documentation

Welcome to the EB tresos AutoCore Generic 8 XCP Stack (ACG8 XCP Stack) product documentation.

This document provides:

- Chapter 2, “Supported features“: list of features supported by the ACG8 XCP Stack
- Chapter 3, “ACG8 XCP Stack release notes“: release notes for the ACG8 XCP Stack modules
- Chapter 4, “ACG8 XCP Stack user guide“: background information and instructions
- Chapter 5, “ACG8 XCP Stack module references“: information about configuration parameters and the application programming interface

2. Supported features

2.1. Overview

This chapter provides an overview of the EB tresos AutoCore Generic 8 XCP Stack and the features that are currently supported.

Section 2.2, “Supported Xcp features” contains an overview of `Xcp` features.

Section 2.3, “Supported XcpR features” contains an overview of `XcpR` features.

2.2. Supported Xcp features

- **Synchronous data acquisition (measurement):** Support for transfer of data packages from slave to master.
- **Synchronous data stimulation (calibration):** Support for transfer of data packages from master to slave.
- **DAQ configuration storing with/without power-up data transfer:** DAQ lists can be configured static at startup or dynamic at runtime by sending commands from master to slave.
- **XCP control commands:** Support for carrying out protocol commands and commands responses.
- **Online calibration:** Support for changing ECU parameterization by sending calibration commands from master to slave.
- **Slave device identification:**
  - Support for master command to get slave identification (type 1 ASAM-MC2 filename without path and extension).
  - Support for retrieving the identification information by calling a user-defined callout for GET_ID, type 1.
- **DAQ timestamps:** Support for sending a timestamp together with data packages in order to get information regarding the exact moment when data has been acquired.
- **Seed & key:** Support for resource protection by using exchange of seed and key algorithm between master and slave.
- **Event prioritization:** Support for assigning a priority to each event and decide the order of processing events considering the configured priority.
- **Support UPLOAD command in block mode:** Master has the possibility to request a data block from the slave by using upload commands.
- **Support for DOWNLOAD command in block mode and DOWNLOAD NEXT command:** Master has the possibility to send a data block that is copied to a specific memory address of the slave.

- **Flash programming:** Master has the possibility to trigger non-volatile memory programming of the slave.
- **Resume mode:** Support for restoring DAQ list configuration from non-volatile memory after startup.
- **Communications bus support:** CAN, CAN-FD, FlexRay, Ethernet
- **Multiple XCP messages in one FlexRay frame:** Support for grouping and transferring multiple XCP messages in one FlexRay frame.
- **Enable/disable the XCP module:** User has the possibility to enable or disable the functionality of XCP module at precompile time or at runtime.
- **DAQ list prioritization:** User can define the order of processing configured DAQ lists by assigning priorities.
- **Distributed handling of event processing:** Support for events processing on different cycles than the main processing cycles of XCP module. A processing function can be generated for each event and scheduled independently by the main processing function of XCP module.
- **Processing of multiple events per Xcp_MainFunction cycle:** Support for processing several events in the same XCP main processing function cycle.
- **Overload indication by setting MSB in the first transmitted packet:** Support for signaling overload situations to the master by sending messages with the most significant bit set.
- **Support for multi-connections:** Possibility to configure different bus interfaces in XCP and change the active connection at runtime by connecting/disconnecting to configured bus interfaces. Only one connection can be active at a time.
- **Support for specialized PDU buffer channels:** Possibility to configure the type of packages to be transmitted or received for each PDU.
- **Support for asynchronous checksum calculation:** Support for processing the checksum calculation asynchronously on a different context.
- **Support for transmission and reception of XCP packages from ISR context:** Support for triggering new transmission requests also from the context of a TxConfirmation and not only from the context of the main processing function. Support for unpacking received packages also in the context of RxIndication.
- **Support for memory access callout functions:** Support for configurable user-defined callout functions for read/write accesses to user memory. If this feature is enabled, XCP does not access user memory internally, but calls user-defined callout functions for performing the read/write access (e.g. during download, upload, modify bits commands).
- **Support for USER_CMD command:** Support for configurable user-defined commands to be sent from master to slave and the reaction on slave side.
- **Support for dynamic communication channels:** Support for enabling/disabling XCP PDUs at runtime by sending a command from master to slave.
- **Support for processing transmission and reception decoupled from the XCP main processing function:** Support to process the reception and transmission on different scheduled entities than the main processing function of the module.

- **Support for XCP service wrapper:** Support for generation of an XCP wrapper that allows you to disable the XCP functionality at run-time. This support depends on a value that is returned from a user-defined callout function.
- **Support for FlexRay hardware buffer configuration:** Support for configuring dedicated buffers in an XCP on FlexRay slave.
- **Support for FlexRay buffer activation and deactivation:** Support for activating or deactivating specific buffers for EV and SERV in an XCP on FlexRay slave.
- **Support for FlexRay buffer assignment to a DAQ List:** Support for assigning individual FlexRay buffers to a DAQ List and reading back the assignment in an XCP on FlexRay slave.
- **Support for memory areas protection:** Support for configuring memory areas and access types for the memory accessible by XCP.
- **Support for calibration page switching:** Support for controlling the XCP slave's logical memory layout (segments, pages).

2.3. Supported XcpR features

- **Support for transport layers CAN, CAN-FD, CDD, FlexRay, Ethernet:** Supported transport layers for master and remote buses are CAN, CAN-FD, CDD, FlexRay, and Ethernet. Header adaptation is performed when different transport layers are required for the routing.
- **Support for multiple routing paths:** Support for multiple configurable sources and destinations of XCP messages. The routing can be done based on pre-configuration for 1:1 mapping between source and destination or at runtime, based on CONNECT request when one source is mapped to two destinations.
- **Support for connection handling:** Support for routing XCP messages to an XCP instance as long as a connection is active (starting an accepted CONNECT command until an accepted DISCONNECT command).
- **Support for multiple XCP messages in one frame:** Support for packing multiple XCP messages in one frame when FlexRay or Ethernet is used for the master bus or remote bus.
- **Support for transmission and reception of XCP packages from ISR context:** Support for triggering new transmission requests also from the context of a TxConfirmation and not only from the context of the main processing function. Support for unpacking received packages also in the context of RxIndication.

3. ACG8 XCP Stack release notes

3.1. Overview

This chapter provides the ACG8 XCP Stack product specific release notes. General release notes that are applicable to all products are provided in the EB tresos AutoCore Generic documentation. Refer to the general release notes in addition to the product release notes documented here.

3.2. Scope of the release

3.2.1. Configuration tool

Your release of EB tresos AutoCore is compatible with the release of the EB tresos Studio configuration tool:

- EB tresos Studio: 28.2.0 b211016-0103

3.2.2. AUTOSAR modules

The following table lists the AUTOSAR modules that are part of this ACG8 XCP Stack release.

Module name	AUTOSAR version and revision	SWS version and revision	Module version	Supplier
Xcp	4.0.3 []	2.0.0 [0]	2.12.2	Elektrobit Automotive GmbH

Table 3.1. Hardware-Independent Modules specified by the AUTOSAR standard

3.2.3. EB (Elektrobit) modules

The following table lists all modules which are part of this release but are not specified by the AUTOSAR standard. These modules include tooling developed by EB or they may hold files shared by all other modules.

Module name	Module version	Supplier
XcpR	1.1.1	Elektrobit Automotive GmbH

Table 3.2. Modules not specified by the AUTOSAR standard

3.2.4. MCAL modules and EB tresos AutoCore OS

For information about MCAL modules and OS, refer to the respective documentation, which is available as PDF at `$TRESOS_BASE/doc/3.0_EB_tresos_AutoCore_OS` and `$TRESOS_BASE/doc/5.0_MCAL_modules`[1]. It is also available in the online help in EB tresos Studio. Browse to the folders `EB tresos AutoCore OS` and `MCAL modules`.

3.3. Module release notes

3.3.1. Xcp module release notes

- AUTOSAR R4.0 Rev 3
- AUTOSAR SWS document version: 2.0.0
- Module version: 2.12.2.B466224
- Supplier: Elektrobit Automotive GmbH

3.3.1.1. Change log

This chapter lists the changes between different versions.

Module version 2.12.2
2021-10-27

- ASCXCP-1111 Fixed known issue: Invalid and unused BswVariableAccess generated in the BSWMD when BSW distribution is not enabled
- ASCXCP-1040 Fixed known issue: ODTs are not cleared for DAQ lists that were not selected before storing to non-volatile memory
- ASCXCP-1115 Fixed known issue: Xcp sends wrong data for events with direction DAQ when bit offset is used

Module version 2.12.1
2021-08-06

[1] `$TRESOS_BASE` is the location at which you installed EB tresos Studio.

- ASCXCP-1096 Fixed known issue: Xcp does not compile because of missing memory section
- ASCXCP-1093 Fixed known issue: Xcp does not compile if BSW distribution and RAM optimizations are both enabled
- ASCXCP-1099 Fixed known issue: Xcp fails to generate when Asynchronous Build Checksum Support is enabled
- ASCXCP-1102 Fixed known issue: Xcp generates a warning when Basic Software Distribution is enabled and multiple connections are configured
- Added support for closing socket connections during deactivation of Xcp

Module version 2.12.0
2021-06-25

- ASCXCP-1032 Fixed known issue: Xcp might not compile if memory mapping is used
- Optimized RAM consumption when a large number of ODT entries are used
- ASCXCP-1066 Fixed known issue: Xcp main functions are not automatically mapped to RTE and BSW events
- ASCXCP-1059 Fixed known issue: Xcp corrupts stimulated data if ODT mirror size is larger than 65535 bytes
- Added BSW Distribution support

Module version 2.11.11
2021-03-05

- Added support for asynchronous processing of calibration commands

Module version 2.11.10
2020-10-23

- ASCXCP-976 Fixed known issue: Field FlxLPDU_ID of Xcp_FlxBufType uses the wrong data type

Module version 2.11.9
2020-06-19

- ASCXCP-872 Fixed known issue: Predefined DAQ lists with lower priorities are considered first if SET_-DAQ_LIST_MODE is not used

- ASCXCP-874 Fixed known issue: Out-of-bounds access occurs when multiple PDUs are packed in one frame on Flexray
- ASCXCP-881 Fixed known issue: CAN-IDs do not have bit 30 set when CAN-FD connections are used
- ASCXCP-880 Fixed known issue: Command packets are rejected if XcpMaxCto or XcpMaxCtoPgm are less than XcpCanFdMaxDlc
- ASCXCP-895 Fixed known issue: Xcp might send wrong data if multiple connections and multiple transmission PDU's are configured

Module version 2.11.8
2020-02-21

- ASCXCP-829 Fixed known issue: Wrong memory section mappings for internal variables Xcp_DaqIdListConst and Xcp_DaqIdList
- Added support for transmission and reception of Xcp messages over a CDD connection.
- Added support for configurable PDU attributes per connection
- ASCXCP-846 Fixed known issue: DOWNLOAD command is rejected with ERR_CMD_SYNTAX if XcpMaxCtoPgm is larger than XcpMaxCto
- ASCXCP-845 Fixed known issue: Xcp can sporadically use a wrong PDU length for the received packages when reception is processed from multiple contexts
- ASCXCP-839 Fixed known issue: Uncompilable code when no Xcp FrIfTxPdu with FrIfImmediate = FALSE exists
- ASCXCP-837 Fixed known issue: Incorrect validation of the configuration parameter XcpFlxBufMaxLenInitValue
- ASCXCP-854 Fixed known issue: Xcp executes user commands with response length exceeding XcpMaxCto
- ASCXCP-861 Fixed known issue: Out of bounds access when transport layer commands are enabled
- ASCXCP-868 Fixed known issue: The second Xcp message from a Flexray frame containing only 2 messages is discarded if its size is less than 4 bytes and 32 bit packet alignment is used

Module version 2.11.7
2019-10-11

- ASCXCP-791 Fixed known issue: XCP always sends TRANSFER_MODE set to 0 in the positive response for the GET_ID command.
- ASCXCP-783 Fixed known issue: CAN identifier in the response of GET_SLAVE_ID and GET_DAQ_ID commands has a wrong format on big-endian architecture.

- Added support for retrieving the identification information by calling a user defined callout for GET_ID, type 1.

Module version 2.11.6
2019-06-14

- ASCXCP-782 Fixed known issue: Linker error may occur because local static constant Xcp_TxPduData-Handlers is not mapped to a memory section.
- Implemented support for Ethernet specific command GET_SLAVE_ID.
- ASCXCP-890 Fixed known issue: CanFD fill bytes are added for different connections which are not CanFD or Can

Module version 2.11.5
2019-02-15

- ASCXCP-765 Fixed known issue: Xcp connections over Ethernet with UDP IPv4 fail when one IPv6 connection is also present.

Module version 2.11.4
2018-10-26

- ASCXCP-753 Fixed known issue: Xcp ignores any incoming message when a FlexRay connection is used, multiple connections are configured and FlexRay reports invalid time information.
- ASCXCP-757 Fixed known issue: Xcp behaves erroneously when FlexRay and other connections are configured.
- XCP slave no longer sends an empty DTO when processing an ODT that has no ODT entries or only OTD entries with the size 0.
- ASCXCP-758 Fixed known issue: Xcp slave performs an illegal memory access when receiving a STIM packet and the DAQ lists are only partially configured.

Module version 2.11.3
2018-06-22

- Added support for SET_MTA endianness.
- Added support for Xcp Memory Access Areas Callout.
- ASCXCP-713 Fixed known issue: WRITE_DAQ_MULTIPLE command fails on big endian platforms when BIT_OFFSET is different from 0.

- ASCXCP-719 Fixed known issue: Compilation error is reported in Xcp_Callouts.c for Xcp_ApplGetSeed function.
- Changed naming of Xcp memory sections to comply with AUTOSAR specifications of memory mapping.

Module version 2.11.2
2018-02-16

- ASCXCP-629 Fixed known issue: Compiler error is issued form Xcp callouts due to undefined symbols.
- ASCXCP-642 Fixed known issue: Xcp slave on Ethernet transmits wrong Counter and Length messages on Big Endian architecture.
- ASCXCP-654 Fixed known issue: Xcp generates an invalid connection configuration if the referenced socket connection is invalid.
- ASCXCP-659 Fixed known issue: Automatic generation of Dem event parameters used by XCP fails when the Service Needs wizard runs.
- ASCXCP-670 Fixed known issue: Xcp performs an out-of-bounds access when Ethernet UDP connection is used.
- ASCXCP-692 Fixed known issue: Xcp slave extracts wrong length from the messages received on Ethernet on big-endian architecture.
- ASCXCP-671 Fixed known issue: EB copyright information is missing in Xcp module.
- ASCXCP-696 Fixed known issue: Xcp performes out of bound access when FLX_ASSIGN command is called with XCP_PACKET_TYPE != 0 and FLX_BUF invalid.

Module version 2.11.1
2017-09-22

- Implemented WRITE_DAQ_MULTIPLE command.
- ASCXCP-581 Fixed known issue: Xcp dedicated FlexRay buffer is not usable when Xcp master send the FLX_ASSIGN command with MAX_FLX_LEN_BUF_x being less than MAX_DTO/MAX_DTO - 1.
- Development errors XCP_E_PDU_BUFFER_LOCKED, XCP_RESP_CTO_QUEUE_FULL ,XCP_E_PDU_BUFFER_FULL, XCP_E_PDU_LOST, XCP_ERR_STIMULATION_DATA_LOST are treated as production errors.
- XCP on FlexRay support for transmission also with immediate buffer access.
- XCP on FlexRay support for transmission in static segment of FlexRay frame.
- ASCXCP-592 Fixed known issue: PROGRAM command post increment MTA address only when the command response is XCP_APPL_OK.
- Added support for FlexRay Sequence Correction.

- Internal module improvement. This module version update does not affect module functionality.
- ASCXCP-615 Fixed known issue: Automatic generation of Dem event IDs used by XCP fails when the Service Needs wizard runs.

Module version 2.11.0
2017-03-24

- Improved the memory consumption by sorting members in structures. Change affects backward compatibility. In case you have enabled the support for storing the DAQ lists in non-volatile (NV) memory (XcpStoreDaqSupported), after updating Xcp to the new version, please ensure that NvM has dynamic configuration support (NvMDynamicConfiguration) enabled and increment the compiled configuration ID (NvMCompiledConfigId). Please note that it is highly likely that the block length of NVM_BLOCK_XCP_DAQ_LISTS must be updated with the new compiled length of Xcp_DaqLists.
- ASCXCP-525 Fixed known issue: Xcp slave could wrongly disconnect from the connection with the master.
- Implemented configurable accesses to protected memory.
- ASCXCP-532 Fix known issue: Xcp user commands are wrongfully rejected if their subcommand codes do not match their index in configuration.
- ASCXCP-548 Fixed known issue: XCP over CAN-FD does not support PDU routing table.
- ASCXCP-547 Fixed known issue: Invalid length of Xcp packages could be considered during FlexRay buffers reconfiguration.
- Implemented support for Xcp_SetTransmissionMode() API. Improved Xcp reaction in case of communication failures.
- ASCXCP-545 Fixed known issue: Xcp could perform a NULL_PTR access when a disconnect happens during an ongoing event processing.
- ASCXCP-557 Fixed known issue: Incompatibility between Xcp and SoAd for Xcp over Ethernet (UDP).

Module version 2.10.0
2016-11-04

- Adapted resource file for the scheduling of main functions to the split of `IpduM_MainFunction()` into `IpduM_MainFunctionRx()` and `IpduM_MainFunctionTx()`.
- ASCXCP-474 Fixed known issue: Wrong header is used for Xcp messages when concatenation of multiple XCP messages in one FlexRay frame is enabled and packet alignment is bigger than 1 byte
- Implemented support for XCP over CAN-FD
- ASCXCP-497 Fixed known issue: Wrong address is accessed by MODIFY_BITS command if Address Translation Callout Support is enabled
- Implemented support for SHORT_DOWNLOAD and DOWNLOAD_MAX commands.

- Implemented support for FlexRay transport layer commands

Module version 2.9.0
2016-05-25

- Updated Bus Timeout detection sequence
- ASCXCP-456 Fixed known issue: XCP on FlexRay does not append an empty header to the actual data when concatenation of multiple messages is enabled

Module version 2.8.0
2016-02-09

- Internal module improvement. This module version update does not affect module functionality.
- ASCXCP-435 Fixed known issue: Xcp does not compile if legacy symbolic names are not provided by DEM
- Implemented Xcp Service Wrapper.
- ASCXCP-441 Fixed known issue: Xcp connection is not possible when the events are processed in their main function faster than the Xcp general main function
- ASCXCP-448 Fixed known issue: Sporadically, Xcp slave could send the same package twice to the master

Module version 2.7.0
2015-11-06

- ASCXCP-413 Fixed known issue: Automatic data transfer after initialization in RESUME mode fails if multiple connections are used.
- ASCXCP-418 Fixed known issue: Missing preprocessor statement in Xcp_Callouts.c template
- ASCXCP-417 Fixed known issue: Communication between XCP master and slave fails if notifications from lower layer use the PduId of upper layer
- Added support for processing reception from RxIndication.
- ASCXCP-424 Fixed known issue: Wrong user command is executed if the configured user commands are not ordered by the sub-command code
- ASCXCP-421 Fixed known issue: Communication on FlexRay fails if multiple connections are used and packing of multiple PDUs is not enabled for all of them.

Module version 2.6.0
2015-06-22

- Added support for triggering transmission from TxConfirmation.
- ASCXCP-393 Fixed known issue: Returned value of Read and Write memory callouts are not evaluated
- ASCXCP-390 Fixed known issue: A command may be lost if the Xcp module is reactivated.
- Added support for User commands.
- Added support for dynamic PDUs and independent scheduled communication functions.
- Added support for processing reception from RxIndication.

Module version 2.5.1
2015-02-18

- ASCXCP-372 Fixed known issue: Connection command is rejected with error code ERR_CMD_SYNTAX
- Added support for XCP multiple connections.
- Added support for specialized PDU buffer channels.
- Added support for multiple PDUs, per connection, for all supported connection types(CAN, FlexRay and Ethernet).
- Added support for asynchronous Checksum calculation.

Module version 2.5.0
2014-10-02

- Added support for multiple XCP messages in one frame
- Introduce user callout functions for memory accesses of XCP commands.
- Added support for multiple PDUs (per connection) (for CAN and FlexRay).
- Added control functionality for Xcp module's state.
- Added support for DaqList prioritization.
- Added support for distributed handling of event processing.
- Added support for processing multiple events per Xcp_MainFunction.
- Added support for indicating an overload situation by setting the Most Significant Bit (MSB) of the PID of the next successfully transmitted packet.

Module version 2.4.0
2014-04-25

Module version 2.3.1
2014-02-14

- ASCXCP-317 Fixed known issue: Xcp can cause an Os exception during DAQ sampling if it is running on a platform where misaligned data is not handled
- ASCXCP-319 Fixed known issue: Parameter `XcpUserHeader` from the Xcp recommended configurations might cause a compilation error

Module version 2.3.0
2013-11-15

- ASCXCP-253 Fixed known issue: Xcp allows configuration of DAQ lists and event channels with direction STIM even if the STIM support is not available and will then ignore STIM packets from the master
- ASCXCP-256 Fixed known issue: Sampled DAQ list data may be inconsistent if the Xcp main function is interrupted during sampling
- ASCXCP-276 Fixed known issue: When processing an ODT, Xcp fails to go to the next ODT when it finds an element with size `0`
- ASCXCP-277 Fixed known issue: When processing a DAQ list, Xcp fails to go to the next DAQ list when the first entry of the first ODT has size `0`
- ASCXCP-273 Fixed known issue: Xcp will stop responding when a Tx confirmation is lost
- ASCXCP-298 Fixed known issue: Generator throws an error if `XcpMaxOdt` or `XcpMaxOdtEntries` does not match the number of ODTs or ODT entries of a predefined DAQ list
- Added support for flash programming
- ASCXCP-306 Fixed known issue: `DISCONNECT` and `SYNCH` commands can be lost or executed twice

Module version 2.2.0
2013-06-14

- Implemented `MODIFY_BITS` command
- ASCXCP-188 Fixed known issue: No DAQ list or the wrong DAQ list may be sampled if more than 255 DAQ lists are configured and a list with a number greater than 255 shall be sampled
- Added support for XCP over Ethernet
- ASCXCP-238 Fixed known issue: Xcp may erroneously reject a `WRITE_DAQ` command for a DAQ list
- ASCXCP-228 Fixed known issue: `GET_DAQ_LIST_MODE` positively responds to a request for a DAQ list which is not configured
- ASCXCP-207 Fixed known issue: The command `ALLOC_ODT_ENTRY` is erroneously accepted and handled if the master requests too many ODT entries

- ASCXCP-243 Fixed known issue: Commands `ALLOC_ODT` or `ALLOC_ODT_ENTRY` may abort with the error `ERR_MEMORY_OVERFLOW`
- ASCXCP-248 Fixed known issue: Xcp may fail or lose packets during resume mode if the communication interface is not in full communication mode

Module version 2.1.0
2013-01-23

- ASCXCP-110 Fixed known issue: The `EV_DAQ_OVERLOAD` event packet is not transmitted if an `OVERLOAD` situation occurs
- ASCXCP-153 Fixed known issue: During allocation of dynamic data acquisition (DAQ) lists, the command `ALLOC_ODT_ENTRY` may abort with the error `ERR_MEMORY_OVERFLOW`
- ASCXCP-152 Fixed known issue: Xcp slave silently ignores requests if the request length is greater than `MAX_CTO`
- ASCXCP-163 Fixed known issue: The Xcp module does not reject commands that are too short which may result in undefined behavior
- ASCXCP-167 Fixed known issue: The Xcp module does not compile if you enable DAQ support but neither configure any data acquisition (DAQ) lists nor any object descriptor table (ODT) for any DAQ list
- ASCXCP-180 Fixed known issue: The command `GET_DAQ_ID` returns a wrong CAN ID for DAQ lists with direction DAQ
- ASCXCP-86 Fixed known issue: The Xcp slave executes a requested command even if it will not be able to send a response
- ASCXCP-181 Fixed known issue: `GET_SLAVE_ID` and `GET_DAQ_ID` return the wrong CAN ID for command and stimulation packets if extended CAN IDs are used
- ASCXCP-208 Fixed known issue: Incorrect development error reported by the Rx indication functions
- Updated to AUTOSAR 4.0 Rev 3

Module version 1.8.0
2012-09-25

- ASCXCP-89 Fixed known issue: The Xcp slave will stop replying to commands (except `DISCONNECT`) if a Tx confirmation is not received
- Added API ID and development error checks to API function `Xcp_SetEvent()`
- ASCXCP-103 Fixed known issue: Xcp slave disconnects from the master when transmission of a message from the slave to the master fails
- ASCXCP-105 Fixed known issue: The Xcp module cannot communicate with a master using a FlexRay communication bus

- Removed comment from wrongfully commented code
- ASCXCP-118 Fixed known issue: `XcpOdtCount` parameter may have the value `0` if the config type is `Dynamic`, which results in invalid generated code
- ASCXCP-112 Fixed known issue: Invalid memory access when handling an event for DAQ/STIM lists
- ASCXCP-128 Fixed known issue: For one specific configuration, `Xcp_TransmitProcessor.c` file does not compile
- Added callout function for address calculation/translation
- Added support for DAQ lists storage (with or without resume mode)
- ASCXCP-143 Fixed known issue: The function `Xcp_ApplBuildChecksum()` is not generated in the template file `Xcp_Callouts.c`
- Changed name of parameter `XcpEventChannelTriggeredDaqListRef` to `XcpEventChannel-TrigDaqListRef` in order to adhere to the 32 character short name restrictions of AUTOSAR 3.1
- Changed name of parameter `XcpRetryFailedReportToDemDetErrorId` to `XcpRetryFailed-DemDetErrorId` in order to adhere to the 32 character short name restrictions of AUTOSAR 3.1
- ASCXCP-123 Fixed known issue: Tx messages may be lost when using the FlexRay bus
- Improved usability of function `Xcp_SetEvent()`
- ASCXCP-145 Fixed known issue: Configuring too many data acquisition (DAQ) lists or object descriptor tables (ODTs) leads to compiler warnings, errors, or wrong data transfer object (DTO) packets sent on the bus
- Added calibration page initialization and switching
- ASCXCP-155 Fixed known issue: The event `EV_STIM_TIMEOUT` may report an invalid event channel or DAQ list number
- Removed `XCP_APPL_ERR_ACCESS_LOCKED` as return value of the Build Checksum callout function
- Protected the event CTO queue accesses with Schm critical sections

Module version 1.7.0
2012-05-30

- ASCXCP-60 Fixed known issue: Removing compiler warnings generated on Pictus platform
- ASCXCP-51 Fixed known issue: Missing positive response for command `DISCONNECT`
- ASCXCP-68 Fixed known issue: The Xcp accepts the `SYNCH` command without an established connection
- Implemented `BUILD_CHECKSUM` command
- Implemented add support for a user-specific timestamp callout function
- Configuration parameters for predefined, static and dynamic configurable daq list were updated according to AUTOSAR 4.0 Rev 3

- ASCXCP-90 Fixed known issue: Transmission of follow-up DTO frames in the transmit interrupt context is not configurable using EB tresos Studio
- ASCXCP-94 Fixed known issue: `Xcp_SetEvent()` may fail to exit its critical section
- Changed main function to also be callable before module initialization
- ASCXCP-52 Fixed known issue: Initialization not meaningful for variables
- Added support for `UPLOAD` command in block mode
- Added support for `DOWNLOAD` command in block mode
- Added support for automatically triggering event channels by the Xcp main function

Module version 1.6.1
2011-12-20

- EBAXCP-248 Fixed known issue: Missing symbolic name for parameter `XcpEventChannelNumber`

Module version 1.6.0
2011-10-28

- Added support for event CTO packets `EV_DAQ_OVERLOAD` and `EV_STIM_TIMEOUT`
- EBAXCP-224 Fixed known issue: Mixing dynamic DAQ lists and predefined DAQ lists generate non-compilable code
- EBAXCP-227 Fixed known issue: Some further DAQs after `STOP DAQ LIST`
- EBAXCP-228 Fixed known issue: `SET_DAQ_LIST_MODE` with error on two dynamic DAQ Lists with two configured events
- EBAXCP-229 Fixed known issue: Initialization of more dynamic ODTs than configured leads to `OS_Exception()`

Module version 1.5.0
2011-10-14

- EBAXCP-213 Fixed known issue: Incorrect XPath check when DAQ List config is `STATIC`
- EBAXCP-216 Fixed known issue: Slave response with `ERROR_OUT_OF_RANGE` for a DAQ list with direction `BOTH` and `WRITE_DAQ` command is requested
- EBAXCP-219 Fixed known issue: Slave response with `ERR_OUT_OF_RANGE` for an `UPLOAD` command request from master
- EBAXCP-220 Fixed known issue: XCP wrong xpath-check leads to wrong failure for parameter `XcpOdt`

- EBAXCP-217 Fixed known issue: XCP was not responding when STIM resource is `OFF` and `DAQLIST` with direction STIM is configured

Module version 1.4.0
2011-09-02

Module version 1.3.0
2011-07-29

- EBAXCP-164 Fixed known issue: The XCP slave returns wrong error code for DAQ commands, when DAQ resource is not supported
- EBAXCP-184 Fixed known issue: The Xcp slave wrongly calculated the total length of ODT entries in an ODT, during dynamic DAQList configuration

Module version 1.2.0
2011-06-30

- EBAXCP-149 Fixed known issue: Misaligned pointer access causing traps/exceptions
- EBAXCP-129 Fixed known issue: Dynamic setting of DAQ list properties (`SET_DAQ_LIST_MODE`) fails after Stop All command(`START_STOP_SYNCH` with Mode 0)
- EBAXCP-151 Fixed known issue: Start/Stop DAQ Measurement fails when it is done more than `MaxDaqList` times for a particular event
- EBAXCP-153 Fixed known issue: The XCP slave returns wrong error code if `ALLOC_ODT` command is given without a `FREE_DAQ` command
- EBAXCP-146 Fixed known issue: The parameter value of `XcpMainFunctionPeriod` can not be changed because of dependency to uneditable parameters like `XcpTimeoutT7`
- Added support for Synchronous Data Stimulation (STIM)
- Added support for commands: `SET_MTA`, `SHORT_UPLOAD` and `DOWNLOAD`

Module version 1.1.0
2011-03-22

- Added support for basic Xcp On FlexRay functionality

Module version 1.0.0
2011-01-21

- First release of EB tresos AutoCore Xcp module

3.3.1.2. New features

- Support for closing socket connections during deactivation of Xcp

  When Xcp_Control is called with XCP_STATE_INACTIVE all sockets which were previously opened by Xcp will be closed if configuration parameter XcpAutoOpenSoCon is set to TRUE.

- New memory section VAR_CONTROL_API_8

  A new memory section VAR_CONTROL_API_8 was added to Xcp. The only Xcp variable which is mapped to this new memory section is Xcp_ControlStateOfXcpModule.

3.3.1.3. EB-specific enhancements

This chapter lists the enhancements provided by the module.

- Support for connection specific PDU attributes

  Description:

  In order to support the functionality of Xcp Router several configuration parameters have been implemented as part of each connection, under the XcpConnectionCfg container. These parameters are: XcpMaxCto, XcpMaxDto, XcpMaxBS, XcpMaxCtoPgm, XcpMaxBSPgm. Additionally, a new parameter XcpTimestampEnabled has been introduced in order to enable or disable timestamps in DTO packets for each individual connection.

- Support of non-cyclic event channels

  Description:

  The module supports non-cyclic event channels for DAQ lists (i.e. event channels, that are not cyclically triggered by `Xcp_MainFunction()`). It provides the API function `Xcp_SetEvent()` to trigger an event for such a non-cyclic event channel.

  Rationale:

  Old module versions only supported to trigger sampling events via the API function `Xcp_SetEvent()`. Therefore this function was retained for compatibility reasons.

- Support of user-specific timestamps

  Description:

  The module supports user specific timestamps (i.e. timestamps that are not read from the AUTOSAR Os).

Rationale:

In this way you can generate the timestamp according to your needs based on own timer hardware or software that can e.g. be more accurate or produces less overhead than an AUTOSAR Os counter.

- Support for configuration of dynamic DAQ list area size

  Description:

  The module supports the configuration of the size of the memory block which is available for dynamically configurable DAQ lists.

  Rationale:

  In this way you can limit the size the Xcp is allowed to use for the allocation of dynamic DAQ lists without the need to know how much size each DAQ list, ODT and ODT entry consumes.

- Support for Bus Timeout Monitor

  Description:

  A Bus Timeout Monitor is implemented which prevents the Xcp slave to remain stuck while waiting to receive the Tx confirmation. This applies in case the message was already accepted by the underlying communication interface, but it cannot be transmitted anymore via the communication bus.

  Rationale:

  When, for example, a busoff event occurs just after the transmission request is accepted by the underlying communication, the Xcp slave remains stuck into Tx confirmation state or TX trigger state. Thus it prevents the establishment of a new communication with a XCP Master which is not able to send a `DISCONNECT` or `SYNCH` request in the specified use case.

- Error code for GET_SEED Command

  Description:

  Current implementation reports following error code for command GET_SEED, even if not specified by ASAM: - ERR_SEQUENCE 0x29

  Rationale:

  The purpose for adding the error code ERR_SEQUENCE is to verify that the request for seeds follows sequence rules. In case the sequence rules are not followed, an ERR_SEQUENCE shall be signalised, because the requested sequence is invalid. - Valid sequence: The master has to use GET_SEED(Mode=1) in a defined sequence together with GET_SEED(Mode=0). - Invalid sequence: If the master sends a GET_-SEED(Mode=1) directly without a previous GET_SEED(Mode=0), the slave returns an ERR_SEQUENCE as negative response, enqueued in the CTO response queue.

- Error code for SET_CAL_PAGE/GET_CAL_PAGE command

Description:

Current implementation reports following error code for commands SET_CAL_PAGE and GET_CAL_-PAGE, even if not specified by ASAM: - ERR_CMD_UNKNOWN 0x20

Rationale:

The purpose for adding the error code ERR_CMD_UNKNOWN is to make sure that in case the slave does not recognise the command it has received, then it should send to the master a negative response, enqueued in the CTO response queue. Handling this situation is EB specific.

- Error code for GET_SEGMENT_INFO command

Description:

Current implementation reports following error code for command GET_SEGMENT_INFO, even if not specified by ASAM: - ERR_MODE_NOT_VALID 0x27

Rationale:

The purpose for adding the error code ERR_MODE_NOT_VALID is to handle the following situation that may occur: - if the slave receives the GET_SEGMENT_INFO command with a selected page mode invalid, the negative response ERR_MODE_NOT_VALID should be enqueued in the CTO response queue and the packet should be sent to the master in order to signal this situation;

- Error code for GET_PAGE_INFO command

Description:

Current implementation reports following error code for command GET_PAGE_INFO, even if not specified by ASAM: - ERR_OUT_OF_RANGE 0x22

Rationale:

The purpose for adding the error code ERR_OUT_OF_RANGE is to handle the following situation that may occur: - if the slave receives the GET_PAGE_INFO command, with a valid syntax, but with parameter(s) that are out of range the negative response ERR_OUT_OF_RANGE should be enqueued in the CTO response queue and the packet should be sent to the master in order to signal this situation;

- Error code for SET_SEGMENT_MODE/GET_SEGMENT_MODE command

Description:

Current implementation reports following error code for command SET_SEGMENT_MODE and GET_-SEGMENT_MODE, even if not specified by ASAM: - ERR_OUT_OF_RANGE 0x22

Rationale:

The purpose for adding the error code ERR_OUT_OF_RANGE is to handle the following situation that may occur: - if the slave receives the SET_SEGMENT_MODE/GET_SEGMENT_MODE command, with a valid syntax, but with parameter(s) out of range, then the negative response ERR_OUT_OF_RANGE should be enqueued in the CTO response queue and the packet should be sent to the master in order to signal this situation;

- Error code for COPY_CAL_PAGE command

  Description:

  Current implementation reports following error code for command COPY_CAL_PAGE, even if not specified by ASAM: - ERR_WRITE_PROTECTED 0x23

  Rationale:

  The purpose for adding the error code ERR_WRITE_PROTECTED is to handle the following situation that may occur: - if the slave receives the COPY_CAL_PAGE command, and the memory location is write protected, then the negative response ERR_WRITE_PROTECTED should be enqueued in the CTO response queue and the packet should be sent to the master in order to signal this situation;

- Error code for PROGRAM_CLEAR command

  Description:

  Current implementation reports following error code for command PROGRAM_CLEAR, even if not specified by ASAM: - ERR_CMD_UNKNOWN 0x20

  Rationale:

  The purpose for adding the error code ERR_CMD_UNKNOWN is to handle the following situation that may occur: - if the slave receives the PROGRAM_CLEAR command, and command is either unknown or not implemented, then the negative response ERR_CMD_UNKNOWN should be enqueued in the CTO response queue and the packet should be sent to the master in order to signal this situation;

- Error code for PROGRAM_MAX command

  Description:

  Current implementation reports following error code for command PROGRAM_MAX, even if not specified by ASAM: - ERR_ACCESS_DENIED 0x24

  Rationale:

  The purpose for adding the error code ERR_ACCESS_DENIED is to handle the following situation that may occur: - if the slave receives the PROGRAM_MAX command and the memory location is not accessible, then the negative response ERR_ACCESS_DENIED should be enqueued in the CTO response queue and the packet should be sent to the master in order to signal this situation;

3.3.1.4. Deviations

This chapter lists the deviations of the module from the AUTOSAR standard.

- No support for READ_DAQ command

  Description:

  This deviation prevents the master from reading back ODT entry details such as the address of a DAQ element, the size of a DAQ element and the address extension of a DAQ element.

- No support for some non-volatile memory programming commands

  Description:

  The following non-volatile memory programming commands are not supported.

  - `PROGRAM_PREPARE`

    This deviation indicates the begin of a code download as a precondition.

  - `PROGRAM_FORMAT`

    This deviation prevents the master from describing the format of data transfer.

  - `PROGRAM_VERIFY`

    This deviation prevents the master from starting slave internal test routines to check whether the new flash contents fits to the rest of the flash.

- No support for transport layer-specific command

  Description:

  Transport layer-specific command `SET_DAQ_ID` for CAN is not supported in the present implementation.

- No support for alternating display mode

  Description:

  This deviation disable the sampling of a single ODT on each DAQ event.

- No support for service request messages

  Description:

  This deviation prevents the slave from requesting actions to be performed by the master device.

- No support for some asynchronous event packets

  Description:

  Some event packets are not supported in the present release:

- EV_CMD_PENDING
- EV_TIME_SYNC
- EV_SLEEP
- EV_WAKE_UP
- EV_USER
- EV_TRANSPORT

This deviation prevents the slave from reporting some events to the master device.

Rationale:

The following event packets are implemented:

- EV_DAQ_OVERLOAD
- EV_STIM_TIMEOUT
- EV_STORE_DAQ
- EV_CLEAR_DAQ
- EV_STORE_CAL
- EV_RESUME_MODE

Not all of the features implemented in the present release use event packets to indicate events in slave to the master. Therefore the others are not yet implemented.

- Unused parameter `XcpOdtEntryMaxSize`

  Description:

  The parameter Odt Entry Max Size (`XcpOdtEntryMaxSize`) is not being used in the current implementation instead `XcpOdtEntrySizeDaq` for DAQ Lists with direction as DAQ, `XcpOdtEntrySizeStim` for DAQ lists with direction as STIM.

  Rationale:

  The new parameters `XcpOdtEntrySizeDaq`, `XcpOdtEntrySizeStim` included in AUTOSAR SWS version V2.0.0 (R4.0 Rev 3) make the parameters common for all Config types. This causes the parameter `XcpOdtEntryMaxSize` to be unused.

- Initialization check in main function

  Description:

  If the main function is called while the module is not yet initialized, the main function returns immediately without performing any functionality and without raising any Det error. This initialization check is always performed independent of the development error detection setting.

  Rationale:

The Xcp's main function may be called before the module is initialized. This would result in lots of Det errors during startup. Therefore the module's main function does not throw a Det error if the module is not yet initialized and simply returns in this case.

- Clearing DAQ lists from the NV memory before storing them

  Description:

  Storing the DAQ lists in the NV memory requires that a clear should be performed beforehand. See ASAM part I - Overview, chapter 1.1.2.2.1 for the specification that states this. However, you do not clear the DAQ lists, but simply overwrite any existing configuration.

  Rationale:

  There is no need to perform the clear as you always overwrite the entire DAQ lists configuration stored in NV memory regardless of the number of selected DAQ lists.

- Extension of the multiplicity of container `XcpOdt` to `0..*`

  Description:

  The multiplicity of container `XcpOdt` is extended from `1..*` to `0..*`.

  Rationale:

  This deviation allows to configure only the necessary parameters for DAQ lists with `XcpConfigType` as `STATIC`. This especially means that no unnecessary ODT parameters must be configured, as they are set by the master during runtime.

- Compiler warnings caused by possible compiler optimizations

  Description:

  Some configurations lead to the situation that the compiler can optimize code away, e.g. because of constant expressions in conditions. This can lead to compiler warnings.

  Rationale:

  Code optimizations are possible in certain configurations. Instead of explicitly building new code for such configurations, it was chosen to let the compiler make such optimizations implicitly.

- Extension of the multiplicity of container `XcpDaqList` to `0..*`

  Description:

  The multiplicity of container `XcpDaqList` is extended from `1..*` to `0..*`.

  Rationale:

This deviation avoids unnecessary DAQ lists configuration when `XcpConfigType` is `DYNAMIC` and no predefined DAQ list is configured or when the DAQ list support is disabled.

- Extension of the multiplicity of container `XcpEventChannel` to `0..*`

  Description:

  The multiplicity of container `XcpEventChannel` is extended from `1..*` to `0..*`.

  Rationale:

  This deviation avoids unnecessary event channel configurations when the DAQ list support is disabled.

- Extension of the multiplicity of container `XcpEventChannelTriggeredDaqListRef` to `0..*`

  Description:

  The multiplicity of container `XcpEventChannelTriggeredDaqListRef` is extended from `1..*` to `0..*`.

  Rationale:

  This deviation avoids to unnecessary assign events to the DAQ/STIM lists in the configuration, as this could be done via command `SET_DAQ_LIST_MODE` for configurable DAQ/STIM lists.

- VSMD parameters are not alphabetically ordered

  Description:

  Additional vendor-specific parameter definitions, container definitions, and references are not added to the VSMD according to the alphabetical order (rule ecuc_sws_1014).

  Rationale:

  A logical feature-based ordering seems more user-friendly and it is easier to maintain.

- `Xcp_Lcfg.c` and `Xcp_PBcfg.c` files are not provided

  Description:

  Because this implementation does not support link time and post-build time configuration, `Xcp_Lcfg.c` and `Xcp_PBcfg.c` files are not needed.

  Requirements:

  Xcp501

- No support for link time and post-build time configuration

  Description:

This implementation does not support link time and post-build time configuration.

Requirements:

Xcp741, Xcp742

- Header files not included

Description:

Some header files are not included in the Xcp module, because they are not needed.

Requirements:

Xcp502

- `XcpOnCan_Cfg.h` file not provided

Description:

The implementation of the XCP on the CAN module does not provide the header file `XcpOnCan_Cfg.h`, because the CAN Interface module does not need this header file.

Requirements:

Xcp506

- XcpOnFr_Cfg.h file not provided

Description:

The implementation of the XCP on the FlexRay module does not provide the header file `XcpOnFr_Cfg.h`, because the FlexRay Interface module does not need this header file.

Requirements:

Xcp507

- `XcpOnEth_Cfg.h` file not provided

Description:

The implementation of the XCP on the Ethernet module does not provide the header file `XcpOnEth_Cfg.h`, because the Ethernet Interface module does not need this header file.

Requirements:

Xcp508

- Interleaved communication mode not supported

Description:

The Xcp module does not support the Interleaved communication mode.

Requirements:

Xcp710

- Detection of production code errors is not always `ON`

Description:

The detection of production code errors can be switched `OFF`.

Rationale:

EB has decided to allow the detection of code errors to be switched `OFF` for a number of reasons including the fact that the reporting of errors costs a lot of runtime. Also the reporting of failed events could result in undefined event memory status.

Requirements:

Xcp756

- AUTOSAR Debugging not supported

Description:

This implementation does not support the debugging mechanisms as described in the SWS specification.

Requirements:

Xcp759, Xcp760, Xcp762, Xcp764, Xcp765

- Configuration address not stored by the `Xcp_Init()` function

Description:

The Xcp module has been implemented to support only the pre-compile time configuration, therefore the `Xcp_Init()` function configuration address parameter is not used.

Requirements:

Xcp802, Xcp834

- `Xcp_GetVersionInfo` not a macro

Description:

`Xcp_GetVersionInfo` has been implemented as a function.

Requirements:

Xcp810

- `GetElapsedValue` not required

Description:

This Xcp module implementation does not require the `GetElapsedValue` service.

Requirements:

Xcp832

- `FrIf_DisableLPdu` not supported

Description:

The Xcp module does not support transmission disabling or the reception of LPdus.

Requirements:

Xcp833

- `SymbolicName` macro generation for XcpOdt

Description:

XcpOdt's symbolic name macros do not follow the AUTOSAR 4.0 Rev 3 naming scheme. The symbolic name for XcpOdt is implemented as `XcpConf_[Shortname DAQ list container]_[XcpOdt]`.

Rationale:

The generation of symbolic name macros for XcpOdt as specified within the ECU configuration specification would lead to multiple macro redefinitions. This applies if more than one predefined DAQ list is configured, as the XcpOdt's short name cannot be dynamically created depending on its `XcpDaqList` container.

- No detection of missing packets for Ethernet

Description:

For Ethernet, the CTR field of a message is not used to detect missing packets.

- Storing DAQ list configuration in NV memory is not available for Xcp on Ethernet

Description:

For Xcp on Ethernet, the storage of DAQ lists in the NV memory feature is not available. This includes the resume mode.

- No support for multiple XCP frames in one UDP/IP frame

Description:

The current implementation of Xcp On Ethernet that uses the UDP protocol only supports one XCP frame in one UDP/IP frame. This deviation prevents Ethernet communication with multiple frames in one frame.

- `SoAdIf_Transmit` name changed

  Description:

  `SoAdIf_Transmit()` interface name has been changed to `SoAd_IfTransmit()`.

  Rationale:

  SoAd specification v2.0.24 changed that API name.

- No support for functional access mode

  Description:

  This deviation prevents the master to program the slave in functional access mode. Only absolute access mode is available.

- No support for `CAN_ID_MASTER_INCREMENTAL` feature

  Description:

  This deviation will prevent the XCP master to send the last request of the block transfer sequence with `CAN_ID_MASTER+MAX_BS(_PGM)-1`.

- No support for all features specified in ASAM MCD-1 (XCP) version 1.2

  Description:

  The protocol version in the generated A2L file is set to 1.2 due to the extensions and new parameters for CAN-FD. The XCP implementation is based on protocol version 1.1 and includes only the extensions related to CAN-FD from the protocol version 1.2. All other new functionalities specified in version 1.2 of the protocol are not supported.

3.3.1.5. Limitations

This chapter lists the limitations of the module. Refer to the module references chapter *Integration notes*, subsection *Integration requirements* for requirements on integrating this module.

- The memory access areas callout feature is not available if the Bsw Distribution feature is enabled

  Description:

If the Bsw Distribution feature is enabled via the BswDistribution container then the memory access areas callout feature cannot be enabled anymore.

Rationale:

Without an asynchronous mechanism, the memory access areas callout feature cannot be used by an integrator to perform cross-core data access. Currently, the memory access areas callout is synchronously called by the Xcp.

- Implementation-specific parameter range limitations

  Description:

  Parameter `XcpEventChannelNumber`: range limited from 0..65535 to 0..65534.

  Rationale:

  This limitation allows for a more efficient implementation.

- Time stamp not supported for ODT consistency

  Description:

  Time stamping of the sampled data is not supported when the consistency of the event channel is ODT.

  Rationale:

  In Xcp, the time stamp is available for each DAQ list and indicates the time at which the ODTs in a DAQ list is sampled. Since in ODT consistency, the ODTs in a DAQ list are not sampled at the same time, the time stamp does not indicate the time at which the ODTs are sampled.

- Implementation-specific parameter range limitations

  Description:

  Parameter `XcpEventChannelMaxDaqList`: range limited from 0..255 to 1..255.

  Rationale:

  This limitation ensures that there is a minimum of one DAQ list associated with the event channel. As, without a DAQ list associated to an event channel, it has no effect.

- Implementation specific parameter range limitations

  Description:

  Parameter `XcpMaxOdtEntries`: range limited from 0..255 to 1..255.

  Rationale:

This limitation ensures that there is minimum one ODT Entry for each ODT configured. As, without an ODT Entry, a configured ODT has no effect.

- Implementation-specific parameter range limitations

  Description:

  Parameter `XcpMaxOdt`: range limited from 0..252 to 1..252.

  Rationale:

  A DAQ list must contain at least 1 ODT to be usable for data acquisition or stimulation.

- Address granularity (AG)

  Description:

  Current implementation supports only address granularity of type `BYTE`.

- Granularity for size of an ODT Entry

  Description:

  Current implementation supports only granularity of type `BYTE` for the size of an ODT entry.

- Implementation-specific parameter range limitations

  Description:

  Parameter `XcpOdtEntriesCount`: range limited from 0..255 to 1..255.

  Rationale:

  An ODT must contain at least one ODT entry to be usable for data acquisition or stimulation.

- Implementation-specific range limitations

  Description:

  If the the storing DAQ lists in non-volatile (NV) memory functionality is enabled then the maximum size (in bytes) possible to be allocated to hold the DAQ lists configuration is limited to 65535.

  Rationale:

  - If the storage in NV memory of DAQ lists is enabled, a NvM block has the size limited to 65535.
  - To avoid any possible alignment issues, the limit of the size is 65532 (which is a multiple of 4).

- XCP_EV_CLEAR_DAQ and XCP_EV_STORE_DAQ events not supported in a BSW distributed context

  Description:

Xcp does not support the XCP_EV_CLEAR_DAQ and XCP_EV_STORE_DAQ events if the BSW distribution feature is enabled.

3.3.1.6. Open-source software

Xcp does not use open-source software.

3.3.2. XcpR module release notes

- Module version: 1.1.1.B466224
- Supplier: Elektrobit Automotive GmbH

3.3.2.1. Change log

This chapter lists the changes between different versions.

Module version 1.1.1
2021-10-27

- ASCXCPR-131 Fixed known issue: Linker error may occur due to missing memory mapping of the internal function declaration.

Module version 1.1.0
2021-06-25

- ASCXCPR-112 Fixed known issue: XcpR_MainFunction() cycle is hard-coded to 1 millisecond.
- Added support for XcpR connections over Ethernet.

Module version 1.0.2
2021-03-05

- ASCXCPR-77 Fixed known issue: XcpR fails to compile if the TriggerTransmit API is enabled and Det functionality is disabled.
- ASCXCPR-80 Fixed known issue: XcpR sends EV_SESSION_TERMINATED events to master Xcp using a wrong PduId.

- ASCXCPR-86 Fixed known issue: XcpR routes data to the wrong PDU ID.
- ASCXCPR-89 Fixed known issue: XcpR incorrectly disables calls to Det_ReportRuntimeError() if development error detection is disabled.
- ASCXCPR-83 Fixed known issue: XcpR reports a DET error when processing the last header of a Flexray frame with multiple messages.
- ASCXCPR-88 Fixed known issue: XcpR might trigger an autonomous disconnect with one MainFunction cycle earlier than configured.

Module version 1.0.1
2020-10-23

- ASCXCPR-60 Fixed known issue: XcpR forwards a wrong trigger transmit pointer to the local slave Xcp.

Module version 1.0.0
2020-06-19

- Implemented XcpR module for initial release.

3.3.2.2. New features

- No new features have been added since the last release.

3.3.2.3. EB-specific enhancements

This module is not part of the AUTOSAR specification.

3.3.2.4. Deviations

This module is not part of the AUTOSAR specification.

3.3.2.5. Limitations

This chapter lists the limitations of the module. Refer to the module references chapter *Integration notes*, subsection *Integration requirements* for requirements on integrating this module.

- Currently no limitations are known in the XcpR module.

3.3.2.6. Open-source software

`XcpR` does not use open-source software.

4. ACG8 XCP Stack user guide

4.1. Overview

This user guide describes the concepts and the configuration of the modules:

- Universal Calibration Protocol (`Xcp`)
- Xcp Router (`XcpR`)

This user guide is intended for readers who have good knowledge of AUTOSAR and about the purpose of the EB tresos AutoCore Generic 8 XCP Stack modules.

4.2. Background information

This chapter describes the basic concepts of the EB tresos AutoCore Generic 8 XCP Stack.

Additional background information is available in the module-specific user guides:

- Section 4.3, “Xcp module user guide” for the `Xcp` module
- Section 4.4, “XcpR module user guide” for the `XcpR` module

4.2.1. Purpose of the ACG8 XCP Stack

4.3. Xcp module user guide

4.3.1. Overview

This user guide describes the `Xcp` module. From this user guide you learn about the basic functionality of the `Xcp`. You also learn which related modules are necessary to configure the `Xcp` module. The `Xcp` module reference provides further information on how to configure the `Xcp` itself.

Note that this user guide is intended for readers who have good knowledge of AUTOSAR and about the purpose of the `Xcp`. The information provided here helps you to integrate the `Xcp` in your AUTOSAR project.

- Section 4.3.2, “Background information” provides an overview of the basic functionality of the Xcp.
- Section 4.3.3, “Configuring Xcp” provides information on related modules that are needed in order to configure the Xcp.
- Section 4.3.4, “Xcp integration notes” provides information on how to integrate the Xcp into your project.
- For details on how to configure the Xcp itself, see the parameter descriptions provided in the Xcp module reference Chapter 5, “ACG8 XCP Stack module references“.

4.3.2. Background information

XCP is a protocol defined by the Association for Standardization of Automation and Measuring Systems (ASAM) for tooling with the purpose of measurement and calibration of ECUs. XCP is a further development of the existing CCP standard that was developed for use over CAN networks. The separation between the protocol layer and the transport layer means that XCP is designed to be used over several network types including CAN and FlexRay. Figure 4.1, “Xcp in the AUTOSAR layered model” shows the location of the Xcp and associated modules in the context of the AUTOSAR layered model.

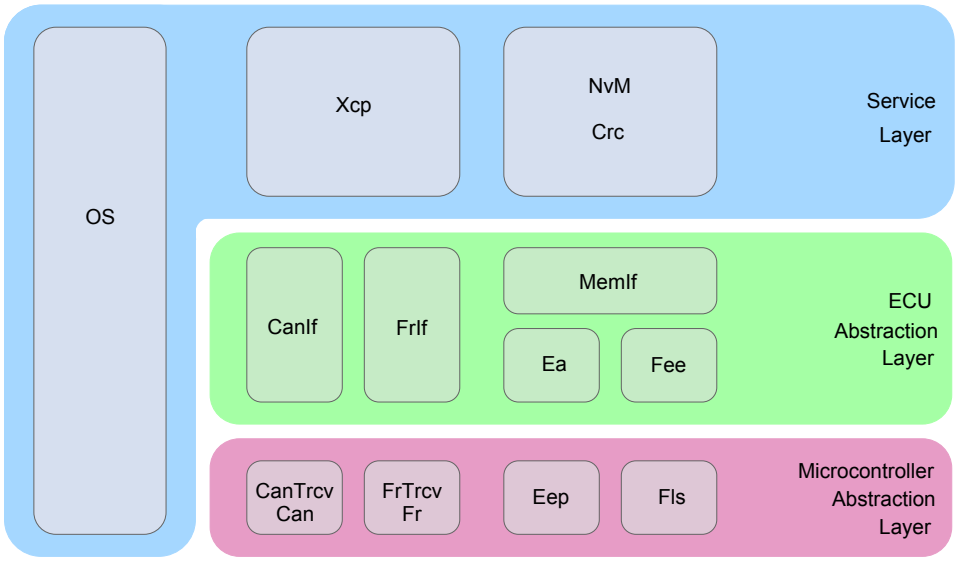

Figure 4.1. Xcp in the AUTOSAR layered model

At the protocol layer, XCP defines functions for measuring, stimulating, calibrating and flashing an ECU. This works in a master-slave manner; a test tool operates as the XCP master while the ECU takes the place of the Xcp slave. AUTOSAR defines an XCP module (Xcp) to run as a slave on an ECU. The network layer provides the required functionality to communicate over the desired network. Figure 4.2, “The topology of an XCP network” shows the topological organization of an XCP network.

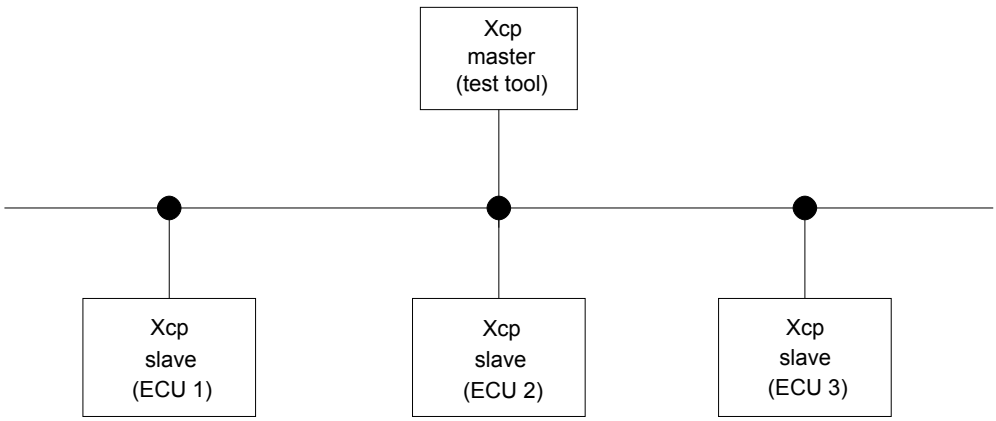

Figure 4.2. The topology of an XCP network

4.3.2.1. Functional overview

An ECU has many parameters that are to be calibrated during development. This calibration is needed to optimize data stored in the ECU. Data measurement is used to observe the behavior of the ECU during the calibration process. Using data acquisition (DAQ) and data stimulation (STIM), ECU data can be synchronously transferred between a test tool and the ECU over a network.

The `Xcp` module supports the following main functions:

- Synchronous data acquisition using DAQ lists, including support for predefined, static, or dynamic DAQ lists
- Synchronous data stimulation using DAQ lists with direction to STIM, including support for predefined, static, or dynamic DAQ lists
- On-line memory calibration using UPLOAD/DOWNLOAD commands
- Calibration data page management
- Flash programming
- Further optional features:
  - Support for using one or more connections via CAN, FlexRay, and Ethernet for the network transport layer.
  - Block mode data transfer to improve performance of the UPLOAD/DOWNLOAD commands. Data can be read out in one cycle rather than polling parameters in turn.
  - Time stamping on data transfer from within the ECU rather than by the test tool, providing more precise timing data.
  - RESUME mode support, with storing of DAQ configuration and power-up data transfer.

4.3.3. Configuring Xcp

To use the Xcp module, you must configure additional modules as outlined below:

- Communication stack: this stack provides the underlying communication required by Xcp. Choose between the CAN stack, FlexRay stack, and the Ethernet stack. For the CAN stack, configure the modules CanIf and Can. For the FlexRay stack, configure the FrIf and Fr modules. For the Ethernet stack, configure the SoAd, TcpIp, EthIf, Eth, and EthTrcv modules.
- EcuC: this module is required for defining the PDUs that are used by the Xcp. For either communication stack, usually you must configure two PDUs within the Xcp module; one for transmission of data and one for receiving data. For the CAN stack, an additional PDU must be configured to receive broadcast messages. In order to configure PDUs in the Xcp module, you must configure the required PDUs first in the EcuC module.
- Dem: for the Xcp, you must configure the Dem events XCP_E_RETRY_FAILED XCP_E_ILLEGAL_MEMORY_ACCESS XCP_E_PDU_LOST XCP_E_PDU_BUFFER_FULL XCP_E_PDU_BUFFER_LOCKED XCP_E_STIMULATION_DATA_LOST XCP_E_ACQUISITION_DATA_LOST XCP_E_RESP_CTO_QUEUE_FULL.

Optionally, the following modules may also be configured:

- Os: an Os counter can be selected for timing calculations that support time stamping. Alternatively, the user can provide a callback function for calculating timestamps.
- Crc: this module provides the standard library routines for calculating CRCs when the optional command BUILD_CHECKSUM is activated. Alternatively, you can provide the CRC calculation for this command in a callback function.
- Memory stack: this stack provides the modules that are needed to support the RESUME mode, i.e. NvM, Crc, MemIf, and underlying modules depending on the type of NV memory used; Ea/Eep for EEPROM or Fee/Fls for flash EEPROM.

  The memory stack may also be used by the application if the calibration page data storage in NV memory is required.

To configure the Xcp, use the information that is provided for each parameter in the module reference: Chapter 5, “ACG8 XCP Stack module references“.

4.3.4. Xcp integration notes

These notes are intended to provide a guide for integrating the Xcp module into your project. The points addressed are specific to issues that may arise due to limitations in the Xcp or in the related modules mentioned in Section 4.3.3, “Configuring Xcp”. Further general integration information is provided in the EB tresos AutoCore documentation.

NOTE **Use of ACG8 XCP Stack in productive environments**

It is assumed that ACG8 XCP Stack is not actively used in productive environments. It is highly recommended to remove XCP before production or to protect the enabling/disabling of the XCP module via mechanisms that satisfy the fact that XCP can manipulate data of the ECU and cause security and safety issues.

4.3.4.1. Event channel processing

In order to sample/bypass RAM data, event channels must be configured. The event channels gather information regarding the job processing list, i.e. DAQ list channels associated to each event channel and event processing speed. The job processing list is given by event consistency and the usage of a DAQ list prescaler. Therefore, it is recommended that the system's timeframe designated for processing event channels should be designed in such a way to be bigger than the designed timeframe for setting event channels. The event channels are usually processed on `Xcp_MainFunction()`. This behavior always occurs for sporadic event channels which are always only set using the `Xcp_SetEvent()` API. However, the channel handling can be distributed over multiple processing runnables. The template for the function name is `Xcp_MainFunction_[EventChannelName]()` and it is used to process the cyclic channel with the corresponding `[EventChannelName]`. For more details, see `XcpEventTriggerMainFunc` and `XcpProcessMultiEventMainFunc` from the Chapter 5, “ACG8 XCP Stack module references“ chapter.

If sporadic event channels are configured, the recommended execution order for the `Xcp` API should be: call the `Xcp_SetEvent()` function before calling `Xcp_MainFunction()`.

To avoid data loss, it is recommended to design the processing of event channels to consider the following recommendations:

- The `Xcp` slaves consider that the event setting represents the starting point for the processing of the entire event channel's job list, i.e. all DTOs are allocated to the event channel. Therefore, it is recommended that the setting of the sampling event minimum latency time has to be higher than the maximum required latency time to complete the processing of the entire event channel's job list. The setting of the sampling event minimum latency time is given either by the lowest time interval for calling `Xcp_SetEvent()`, or by the configured value for `XcpEventChannelTimeCycle`[`XcpEventChannelTimeUnit`]. The setting depends on the event consistency level, which is a time multiple of `XcpMainFunctionPeriod`, and it also depends on the usage of the DAQ list `XcpDaqListPrescaler`.
- To decrease the required amount of time required to process an event channel, and to decrease the number of DTOs available for transmission, it is recommended to use `XcpDaqListPrescaler` to delay the processing of some of the DAQ lists from the event channel job list.
- The configured transmission bandwidth for the `Xcp` module has to be high enough to allow sending of all DTO buffered data before the buffered data is overwritten. To avoid overload events, all sampled DTOs must be sent before they are sampled again.

Caveat: `FREE_DAQ` or `CLEAR_DAQ_LIST` commands will return `ERR_CMD_BUSY` whenever an event main function (`XcpEventTriggerMainFunc` parameter) is running at the time when these commands are executed and that event has DAQ lists started. This is needed in order to prevent the Xcp master from changing the configuration of the DAQ lists while data aquisition/stimulation is ongoing from the event main function context.

4.3.4.2. Data loss signaling between master and slave

The idea of using DAQ list overload or STIM timeout detection is to create a framework where a data loss can be signaled between the master (tool) and the slave (ECU). Two methods are used to signal a data loss event for the following situations:

- STIM timeout mechanism is used to signal the master that not all the STIM packages required to write bypass data were received in the expected time frame until the processing of the event channel is executed;
- Overload mechanism is used to signal that some of the expected DTO packages are lost. These DTO packages are overwritten or not processed at all. The sources for overload might be one of the following:
  - The event channel is set again before the execution of the event channel job list is finished;
  - The event is not executed because an error is detected. This situation applies only for event channels with consistency *EVENT*, and occurs due to the detection of a STIM timeout error, while the event is processed. This is the only situation when the result of data sampling operation depends on the result of a data bypassing operation;
  - DTO data is overwritten because the transmission bandwidth is too small. This behavior affects all sampled DTOs which cannot be send before they are sampled again.

There are two ways of reporting an overload event, which are managed by the value configured for the `XcpSupportForOverloadMSB` configuration parameter:

- Sending overload information that uses an `EV_DAQ_OVERLOAD` package. The event is prepared to be sent upon detection, only once per transmit cycle, even if it is detected for more than one time within one transmission cycle. This is the type of notification that does not give a clear information regarding which data packages are affected by the overload detection.

  If you need more accurate overload information, you can use the next notification type instead:

- Sending the overload information coded into PID's most significant bit.

  The advantages of using this feature are:

  - The overload information is strictly bounded to the DTO package which is affected by the overload detection;
  - This feature disables the sending of `EV_DAQ_OVERLOAD` packages, which increases the data transmit throughput by not preparing and sending additional package/s that contain the overload information.

For more details regarding the configuration parameters, see Chapter 5, “ACG8 XCP Stack module references“.

4.3.4.3. Usage of the RESUME mode feature

In order to use the RESUME mode feature in the Xcp, it is necessary to configure a block in the NvM module. The parameter descriptions provide detailed information on specific values that are needed to configure the NvM block. To find this information, see the module reference Chapter 5, “ACG8 XCP Stack module references“, parameter XcpStoreDaqNvmBlock.

An Xcp callback function must be called when the DAQ list configuration has been written to NV memory on request from the master. This callback function can be configured directly in the NvM as described in the parameter description for XcpStoreDaqNvmBlock.

The NvM block used to store the DAQ configuration is expected to be read during execution of the NvM_ReadAll() function. Therefore, you must ensure that the NvM_ReadAll() function has completed its execution before the EcuM/BswM calls the Xcp_Init() API function during startup. If the expected data was not successfully read from the NV memory, the Xcp initialization provides default DAQ list data.

4.3.4.4. Support for calibration page data management

In effect, the management of calibration page information must be handled by the application.

To support the calibration data page switching function, you must implement several callout functions, which are needed by the Xcp module. To use calibration data page switching, see the module reference Chapter 5, “ACG8 XCP Stack module references“, parameter XcpCalPagSupported, to find further information together with the list of functions that must be implemented.

In addition, calibration data may be stored to non-volatile (NV) memory. To support this feature, you must implement a callout function that writes the calibration data to NV memory when requested by the Xcp master. This callout function may also use the memory stack function in order to achieve this. On completion of writing, a Xcp callback function must be called. For details of the callout and callback functions that support this feature, see the module reference Chapter 5, “ACG8 XCP Stack module references“, parameter XcpCalPagStoreSupported.

4.3.4.5. Support for programming in non-volatile memory

In effect, the flash programming must be handled by the application.

To support the flash programming function, you must implement several callout functions, which are needed by the Xcp module. To use flash programming, see the module reference Chapter 5, “ACG8 XCP Stack module references“, parameter XcpPgmSupported, with a list of functions that must be implemented.

4.3.4.6. Support for multiple connections

Configuring multiple connections is now possible in the Xcp module with the restriction that only one connection is active at one moment in time. The Xcp slave is designed to provide the required mechanisms and functionalities for environments where multiple Xcp masters are connected to the Xcp slave on different communication interfaces. However, the Xcp slave can be connected to only one master at the same time. The following connection types are defined from the Xcp slave point of view:

- Xcp connection over the CAN communication bus.
- Xcp connection over the FlexRay communication bus.
- Xcp connection over TCP/IP (Ethernet communication bus).
- Xcp connection over UDP/IP (Ethernet communication bus).

Because the Xcp module acts at the service layer, it manages PDUs. Therefore the basic element to map an unique PDU channel towards each available connection is to have multiple connections available. In this way, multiple connections of the same type can coexist within the same configuration.

However, only one connection can be active at a time. This restriction is based on the single master - single slave connection topology specified by the ASAM specification. For more details, see the XcpConnectionCfg data container from module reference Chapter 5, “ACG8 XCP Stack module references“.

4.3.4.7. Support for multiple PDUs per connection

The ASAM defines two classes of objects used to achieve exchange data between the client (Xcp master tool) and the server (ECU running the Xcp slave): CTO and DTO. Each of the two mentioned classes is divided by the ASAM specification into subclasses as following:

- Subclasses for CTO:
  - CMD packages represent the CTO packages sent by the client and received by the server.
  - RES/NEG packages represent the CTO packages sent by the server and received by the client, and are triggered as a response for the previous received CMD packages sent by the client.
  - EV/SERV packages represent the CTO packages sent by the server and received by the client, which are triggered by events, and are not necessary related to the processing of an incoming CMD package.
- Subclasses for DTO:
  - STIM packages represent DTO packages sent by the client and received by the server.
  - DAQ packages represent DTO packages sent by the server and received by the client.

NOTE **Restricted number of PDU channels**

1. The number of CMD and RES/NEG PDU channels are restricted to one PDU channel for each type per connection. One exception here: if the connection is over CAN, one more PDU channel can be configured to receive the broadcast `GET_SLAVE_ID` command.
2. In case of EV/SERV, STIM and DAQ subclasses, the only restriction for the number of the PDU channels supporting these types if the enumerated resources are available: there can be more than one PDU channel configured for each type per connection.

From the server point of view, the PDU channels can be configured, as following:

- Any transmitted PDU can be configured to support any combination from the following object subclass types: RES/NEG, EV/SERV, and DAQ.

  If a package is available to be sent, the following rule applies: the server sends the mentioned package on the first available PDU channel which supports the transmission of the subclass object of which the package belongs to. If multiple PDU channels are available, the pending package is sent via the PDU channel which has the smaller configured internal PDU ID, which follows the rule mentioned above.
- Any received PDU can be configured to support any combination from the following object subclass types: CMD and STIM.

  Vice versa, all incoming packages are dropped, which are received on a configured PDU channel which does not support the reception of the subclass object of which the package belongs to. The incoming PDUs are processed in the ascending order of configured internal PDU IDs.

  For more details, see the `XcpTxPduConnectionInfo` and `XcpRxPduConnectionInfo` data containers from the module reference Chapter 5, “ACG8 XCP Stack module references“.

4.3.4.8. Support for packing multiple Xcp PDUs in one frame

The latest version of the `Xcp` supports packing of multiple PDUs in one frame.

The number of packages packed into a frame is variable and depends on how many packages are available for transmission/reception. For the moment, all kinds of packages are packed into the frame, meaning that the tool should be able to unpack both CTOs and DTOs from each one of the configured transmission PDU. The same package packing sequence is accepted to receive PDU packages.

There are though some limitations regarding the incoming packages:

- Only one interleave command is accepted to be packed inside one frame. If there are many more interleaved commands packed into one frame, the first detected interleaved command is accepted and all the others are silently ignored.

- In contrast to the interleave command case, multiple block commands are accepted to be packed inside the incoming frame. The limitation is that if the configured command buffer gets full, the remaining block mode commands are silently ignored.
- There is no limitation regarding the number of packed STIM packages inside the incoming frame: all STIM packages are accepted.

This feature is supported only for FlexRay and Ethernet and it can be individually activated for some of the configured connections which support the feature. For more details, see Chapter 5, “ACG8 XCP Stack module references“.

For the CAN protocol, this feature is not supported because it is not practical to use this feature on the CAN bus due to a PDU length limitation to maximum 8 bytes for this communication protocol.

4.3.4.9. Support for Xcp on Ethernet

If the desired protocol is TCP/IP, configure for each `Xcp` connection in the `SoAd` module a single socket connection. This socket connection shall have mapped all `Xcp`'s Rx and Tx PDUs configured for an `Xcp` connection.

If the UDP/IP protocol is used for the selected `Xcp` connection, connections with two sockets must be provided: one for all mapped `Xcp` Rx PDUs and another one for all mapped `Xcp` Tx PDUs. These connections with two sockets have to belong to the same socket connection group.

For each available `Xcp` over Ethernet connection, the `Xcp` may open the necessary sockets in its initialization routine. However, if you configure the `XcpAutoOpenSoCon` parameter to false, it is possible to open the necessary sockets elsewhere in the application. If `XcpAutoOpenSoCon` is set to true, make sure that `Xcp_Init()` is called after the `SoAd_Init()` in the initialization routine.

If multiple Ethernet connections are configured, it is not allowed to share the same socket connection group between `Xcp` Ethernet connections.

If the UDP/IP protocol is used for the selected `Xcp` connection, `SoAdSocketAutomaticSoConSetup` must be set to false and `XcpAutoOpenSoCon` must be set to true.

Note that if `XcpAutoOpenSoCon` is set to true, then `Xcp` calls `Soad_OpenSoCon()` only in order to open the necessary sockets. `Soad_CloseSoCon()` is only called by `Xcp` as a result of calling `Xcp_Control()` (if parameter `XcpEnableXcpControlApi` is true) with mode XCP_STATE_INACTIVE when the `Xcp` is already in the active state.

4.3.4.10. Support for Xcp on FlexRay sequence correction

In order to enable FlexRay sequence correction, you must enable the `XcpSequenceCorrectionEnabled` parameter.

Sequence correction consists of a 1 byte counter that is sent with each frame. The counter starts with 0 and is post-incremented after each successfully transmitted Xcp packet.

On reception side, the counter that is sent by the Xcp master is ignored as no meaningful reaction was identified in case of counter inconsistencies.

4.3.4.11. Usage of the BUILD_CHECKSUM command

The BUILD_CHECKSUM command represents the type of command which might take a long time to be executed, depending on the memory block size. Therefore, two approaches are implemented and are explained below:

1. The first approach is represented by a synchronous execution of the checksum operation. With this approach, the BUILD_CHECKSUM command is decoded and executed within one cycle of the Xcp_MainFunction() call.

   The first approach is useful if the memory block size is relatively small and the execution time can fit in one Xcp_MainFunction() call cycle, without disturbing the execution of other system tasks.

2. The second approach is represented by an asynchronous execution of the checksum operation. In this case, the BUILD_CHECKSUM execution is split between the following Xcp runnables:

   - Xcp_MainFunction(): within this runnable, the command is decoded, when it is received from the Xcp master and the response package is prepared to be sent back to the Xcp master after the checksum calculation is completed.

     This runnable is probably running from the context of a fast task.

   - Xcp_CrcMainFunction(): within this runnable, the checksum calculation is executed when a pending checksum request is active.

     This runnable is expected to run from the context of a much slower, even preempted task, e.g. a background task.

     If the task that runs the Xcp_CrcMainFunction() is preempted, the SCHM_XCP_EXCLUSIVE_AREA_XCP_INTERNALS must be mapped to an interrupt locking mechanism. For more details, see Section 4.3.4.17, “Xcp critical section mapping and scheduling notes”.

   The second approach is more suitable if a large memory block size is used to calculate the checksum, because the checksum job calculation can be interrupted by other more urgent system tasks.

4.3.4.11.1. Execution time of asynchronous calculation

1. No advantage is achieved if both runnables Xcp_MainFunction() and Xcp_CrcMainFunction() are mapped to the same task.

Use synchronous BUILD_CHECKSUM calculation instead of asynchronous BUILD_CHECKSUM calculation as a better solution than the one to map both runnables to the same task.

2. A greater attention must be given to the overall time which is consumed to complete the BUILD_CHECKSUM command execution, when the asynchronous checksum calculation is configured. The total execution time must not exceed the configured timeouts for the Xcp master.

   An example of a simple formula used to calculate the total time required to finish the asynchronous execution of the BUILD_CHECKSUM command:

   ```
   T_Total=2 * T_(TotalXcp_MainFunction)+ n * T_(Xcp_CrcMainFunction)
   ```

   n represents the required number of Xcp_CrcMainFunction() cycles needed to complete the checksum calculation.

4.3.4.12. Handle ID assignment

According to the AUTOSAR specifications in AUTOSAR_TPS_ECUConfiguration.pdf, the choice of the value for a handle ID is open to the implementation of the providing module. There might be different strategies to optimize the handle ID values and therefore the internal structures of the implementation may have an influence on the choice of the values.

Handle IDs are defined by the module providing the API and are used by the module that calls the API. Handle IDs that are used in callback functions, e.g. Tx confirmation functions or trigger transmit functions, shall be defined by the upper layer module. In the upper layer module the same handle ID shall be used for the Tx confirmation and for the trigger transmit callback functions. That is, the module that receives a transmission request can call the Tx confirmation callback with a different handle ID than the transmission request handle ID. This is a difference to previous releases of AUTOSAR where the Tx confirmation was called with the same handle ID.

The implementation of the Xcp module provides two different mechanisms to retrieve and exchange the handle IDs with the lower layer. By default the approach is the one specified by AUTOSAR, meaning that the PDU IDs that are exchanged with the lower layer are the PDU IDs defined by the Xcp. The second approach allows the usage of an internal routing table in the Xcp. The role of this table is to map PDU IDs from the lower layer to the PDU IDs of the Xcp and all callback functions from the lower layer use the PDU IDs of the lower layer. You can choose between the two options by configuring the parameter XcpSupportForPDURoutingTable. By default this parameter is disabled and no mapping of the PDU IDs is done in the Xcp.

4.3.4.13. Message transmission from slave to master

There are four possibilities to trigger the transmission of messages from the XCP slave to the master:

- From the context of `Xcp_MainFunction`. In this case as long as there is data to be transmitted, messages are packed and sent on the bus based on the main processing cycle of the module.
- From the context of `Xcp_EventMainFunction` if configured. In this case data is acquired based on the event processing function and as long as there is data available, the transmission is triggered from the event processing function and also from the main processing function of the `Xcp` module.
- From the context of `Xcp_<BusIf>TxConfirmation()` if configured, see configuration parameter `XcpPduSupportTxFromTxConfirmation`. In this case data is packed and transmitted from the TxConfirmation of the underlying layer, from the `Xcp_EventMainFunction` if configured, and from `Xcp_MainFunction`.
- From the context of `Xcp_TxMainFunction` if configured, see configuration parameter `XcpSupportComMainFunctions`. In this case an additional main processing function shall be used for transferring data in the `Xcp`, and `Xcp_MainFunctionTx` and `Xcp_MainFunctionRx` are schedulable functions with a configured periodicity.

NOTE **Impact of message transmission on the CPU load**

It is not the job of the `Xcp` module to assure that based on the configuration the transmission of messages is not creating an overhead in the CPU load. As long as the transmission is done from several contexts especially from interrupt context, during integration phase it shall be assured that the time needed for packing and sending messages on the bus does not affect the runtime of the application and it does not increase the CPU load.

4.3.4.14. Message reception from master to slave

There are three possibilities to process the incoming messages from `XCP` master to the slave:

- From the context of `Xcp_MainFunction`. In this case as long as there is data to be received, messages are received on the bus and unpacked based on the main processing cycle of the module.
- From the context of `Xcp_EventMainFunction` if configured. In this case data is acquired based on the event processing function and as long as there is data available, the reception is triggered from the event processing function and also from the main processing function of the `Xcp` module.
- From the context of `Xcp_<BusIf>RxIndication()` if configured, see configuration parameter `XcpPduSupportRxFromRxIndication`. In this case data is unpacked and processed from the `RxIndication` of the underlying layer.
- From the context of `Xcp_RxMainFunction` if configured, see configuration parameter `XcpSupportComMainFunctions`. In this case an additional main processing function shall be used for reception of data in the `Xcp`. `Xcp_MainFunctionTx` and `Xcp_MainFunctionRx` are schedulable functions with a configured periodicity.

NOTE

Impact of message reception on the CPU load

It is not the job of the `Xcp` module to assure that, based on the configuration, the reception of messages does not create overhead in the CPU load. As long as the reception is done from several contexts, especially from an interrupt context, during integration phase it shall be assured that the time needed for receiving and unpacking messages on the bus does not affect the runtime of the application and it does not increase the CPU load.

4.3.4.15. Support for Xcp on CAN-FD

The maximum data length of a CAN message and therefore maximum length of an XCP on CAN message is MAX_DLC = 8. The maximum data length of a XCP on CAN-FD message equals one of the following values i.e, MAX_DLC (`XcpMaxDlc`) = 8, 12, 20, 16, 24, 32, 48, 64.

For CAN-FD, the data length of the XCP packages is determined in two ways:

- If `XcpMaxDlcRequired` (MAX_DLC_REQUIRED) is set, the length of XCP packages to be sent on the bus is given by the configuration of parameter `XcpCanFdMaxDlc`.
- If `XcpMaxDlcRequired` (MAX_DLC_REQUIRED) is not set, the length of XCP packages equals next higher valid valued of CAN-FD DLC.

If you want to use XCP over CAN-FD, you must enable the configuration switch `XcpOnCanFDEnabled` and set the `XcpConnectionInterfaceType` to `XcpConnectionOverCANFD`

You can define the value to be used as fill byte for the transmitted XCP packages by configuring the parameter `XcpCanFdFillValue`.

4.3.4.16. Support for Xcp on CDD

The maximum data length of a CDD message and therefore maximum length of an XCP on CDD message is determined by the bus type represented by the Complex Device Driver.

As there is no standard CDD configuration, the lower layer destination PDU ID needs to be set by configuring the `XcpCddLowerLayerTxPduId` parameter.

If the TriggerTransmit functionality for CDD is required, the `XcpTxPduSupportForCddTriggerTransmit` parameter needs to be set.

If you want to use a XCP connection over CDD, you must enable the configuration switch `XcpOnCddEnabled` and set the `XcpConnectionInterfaceType` to `XcpConnectionOverCDD`

4.3.4.17. Xcp critical section mapping and scheduling notes

The Xcp uses a critical section to protect its internal queues. This behavior is named SCHM_XCP_EXCLUSIVE_AREA_XCP_INTERNALS. Map this to an interrupt locking mechanism, e.g. ALL_INTERRUPT_BLOCKING in the Rte module configuration.

If the Xcp is to be integrated with the memory stack, e.g. to use the RESUME mode feature or for storage of calibration data to NV memory, take care when you configure the scheduling of the main function calls in the Rte module. To ensure that sensitive areas in the Xcp module are not interrupted by the NvM module, schedule the Xcp main function call in the same task as the NvM main function call. Alternatively, design the task schedule to ensure that the Xcp main function is not interrupted by NvM main function calls.

4.3.4.18. Xcp memory mapping notes

In order to use dynamically configurable DAQ lists with the Xcp, the memory section XCP_xxx_SEC_VAR_SAVED_ZONE_UNSPECIFIED must be defined in such a way that the start address of the variable which is placed in this memory section meets all hardware requirements regarding the alignment of pointers to variables.

4.3.4.19. User-defined commands

You can extend the normal functions of the Xcp module with XCP user commands. You can define each user command and provide a callout function to be executed when the command is requested by the master.

The filtering and execution mechanism of a user command follows the same steps as normal commands. A connection must be established between the master and the slave and once a user command is received, the following checks are made:

- The length of the message is checked for being within the valid range.
- The subcommand code has to be configured.
- If it is enabled by configuration, the expected resource has to be unlocked.

If these conditions are fulfilled and a user callout is configured, this user callout is executed. The actions done inside the callout are user-defined. User commands must not be used to implement functions done by other services.

The user-defined callout function must respect the following prototype:

```
FUNC (uint8, XCP_CODE) XXX_UserCallout(
     P2CONST( uint8, AUTOMATIC, XCP_APPL_DATA ) ParamPtr,
     uint8 Length,
     P2VAR( uint8, AUTOMATIC, XCP_APPL_DATA ) SubCmdResponse,
     uint8 RespLength
     )
```

Parameter set:

- Input parameters:
  - `ParamPtr` - represents a pointer to the parameter buffer
  - `Length` - the length of the parameter buffer; this shall be the same value as the one in the configuration
  - `RespLength` - the length of the response; this shall be the same value as the one in the configuration
- Output parameters:
  - `SubCmdResponse` - represents a pointer to the response buffer

Return value:

- If the return of the callout function will be `E_OK`, the master will receive the callout function response

  If any other value than `E_OK` is returned by the callout function, the response will be the error code: `ERR_GENERIC` .

4.3.4.20. Support for dynamic PDU channels

With this feature you can have dedicated `Xcp` communication channels that can be actively used only when lots of data has to be exchanged between master and slave.

This means that specific `Xcp` PDUs configured for DAQ and STIM can be enabled or disabled at runtime. It shall not be possible to configure PDUs used for transmitting command, response, and event packages as dynamic.

As long as this feature is not specified by the ASAM, the request from the master is triggered by the reserved user command subcommand code `0x00`. This means that the subcommand with code `0x00` has to be reserved and configured accordingly:

- Subcommand code: `0x00`
- Subcommand length: 3
- Subcommand response length: 0
- Subcommand user callout: `Xcp_EnableCommunicationChannel`
- Subcommand resource: not fixed, user can select the resource

The command to be sent from master to slave in order to enable or disable PDUs must have the following format: Command_Code SubCommand_Code Direction PduId PduState

- Command code: `0xF1` - user command
- SubCommand code: `0x00` - reserved subcommand code
- PDU direction: `0x00` - transmission; 0x01 - reception

- PduId: configured index for an `Xcp` PDU
- State: state requested for a specific PDU - `0x00` - disabled, `0x01` - enabled

4.3.4.21. Support for independent main functions for processing communication

There are projects where it is needed to process the transmission and reception of PDUs with a higher frequency than the normal main processing function. For example once an event is processed, large amounts of data must be exchanged between master and slave. To exchange the data for stimulation or acquire data based on a different cycle, you can use this feature and you can schedule independent processing functions for processing communication based on a different cycle than the main processing function of the module.

4.3.4.22. Support for deactivation of transmission capabilities

`Xcp_SetTransmissionMode` is used to turn on and off of the TX capabilities of used communication bus channel in XCP module. In order to activate the feature, you must set `XcpSuppressTxSupport` to `true`.

Caveat: parameter `Channel` has a 1 to 1 relationship with an Xcp connection (defined by parameter `XcpConnectionId`).

4.3.4.23. Transmission failure monitoring

Each message accepted by the lower layer for transmission is monitored for a period of time (defined by parameter `XcpTxBusTimeout`) and, during that monitoring time, Xcp slave expects the message to be transmitted successfully, i.e. the `TxConfirmation` is received.

In order to retry the whole message transmission you must configure `XcpTxBusRetry`. If the message timeouts and the number of configured retries have not been reached, the Xcp slave will retry the message transmission process (i.e. call `xxxIf_Transmit()` once again with the same message). It is possible to retry indefinetly if you configure `XcpTxBusRetry` to 255.

If the message timeouts for the number of configured retries, the connection is deemed compromised and the Xcp slave will autonomously disconnect from the Xcp master.

If you have configured event packets (`XcpEventPacketEnabled`), before disconnecting Xcp slave will try to notify the master with the event packet `EV_SESSION_TERMINATED`.

4.3.4.24. Disable Xcp functionality

As XCP module is generally used only during development phase or in the garage, in several cases it is required to disable the functionality during normal operation of an ECU. EB provides two different mechanisms for disabling the XCP functionality:

- A configurable vendor specific interface is available and you have the possibility to enable/disable the XCP functionality by calling this interface. To define the state of XCP you can use the interface `Xcp_Control` that is available depending on the value of the configuration parameter `XcpEnableXcpControlApi`.
- A wrapper can be generated for the XCP module. The caller of the XCP interfaces only calls the wrapper interfaces. The wrapper always checks (by calling a user defined callout) if the XCP module is enabled and if so it calls the XCP interface. If the XCP is disabled the wrapper simply returns without any effect.

  You can generate the wrapper interfaces by enabling configuration parameter `XcpServiceWrapper`. In this case you have to assure that the mandatory user defined callout function is defined as:

  ```
  FUNC(uint8, XCP_APPL_CODE) App_Xcp_Wrapper_IsXcpUsed(void)
  ```

  By enabling this feature XCP will only generate the schedulable entities and timing events for Xcp_Wrapper. Also you have to assure that no XCP interface is configured in any module calling the XCP. Only Xcp_-Wrapper interfaces shall be configured.

  You shall not disable the module during normal operation or while waiting for any callback functions to be triggered. No wrapper interface is generated for callback functions that imply normal operation of the XCP. For example no wrapper is generated for the callback from NvM as it is considered that the module state is not changing from the moment a request to NvM is started until it is completed. If the module is disabled during this time the callback is lost anyway.

4.3.4.25. Support for Xcp FlexRay buffers

With this feature you can:

- Assign an `LPDU_ID` to an XCP-dedicated buffer of a slave via command `FLX_ASSIGN`
- Assign an `XCP_PACKET_TYPE` to a Buffer via command `FLX_ASSIGN`
- Change the length (`MAX_FLX_LEN_BUF_x`) of a Buffer via command `FLX_ASSIGN`
- Activate/deactivate `EV_SERV` packet type via commands `FLX_ACTIVATE/FLX_DEACTIVATE`
- Assign buffers to a specific DAQ list via commands `SET_DAQ_FLX_BUF/GET_DAQ_FLX_BUF`

For simplicity sake, by "Buffer" it is meant a Xcp-dedicated FlexRay buffer of a slave.

In order to activate the feature, Xcp must be on FlexRay (`XcpOnFlexRayEnabled`) and Support for FlexRay Transport Layer Commands (`XcpSupportForFlxTPCommands`) must be enabled

In order to configure the feature, you must consider the following:

- Each Buffer must have an initial `LPDU_ID` assignment (`XcpFrIfLPDURef`). With this assignment the Buffer type is determined (either Tx or Rx)
- Each Buffer configured in XcpFlexrayBufferCfg must be mapped to a Connection Info (either `XcpTxP-duConnectionInfo` or `XcpRxPduConnectionInfo`) depending on the Buffer type

- The LPDU_IDs which are going to be assigned to a Buffer are mapped to a FlexRay frame. That frame must contain only one FrIf PDU (reason is detailed in the limitations chapter of the release notes). That FrIf PDU must point to the same ECUC PDU that the Connection Info points to.
- Each Buffer's initial Packet Type assignment is defined in the associated Connection Info.
- Only one buffer with CMD packet type and only one buffer with RES_ERR packet type is allowed per connection.
- There must be at least two buffers (one Rx and one Tx) fully initialized (LPDU_ID and MAX_FLX_LEN_BUF_x_init parameters must have initial values) and assigned each to a Connection Info that have the CMD and RES_ERR packet types enabled. This ensures that there is an initial working configuration.
- GET_DAQ_FLX_BUF will always return FLX_BUF_FIXED parameter with the value 1. The reason is that the description of this parameter in the ASAM specifications is ambiguous.
- A buffer can be assigned only to one DAQ list at a time if SET_DAQ_FLX_BUF is used. If SET_DAQ_FLX_BUF did not explicitly assign a DAQ list to a buffer, then any DAQ list can use that buffer.
- SET_DAQ_FLX_BUF will initially clear the previous buffers assignment for the issued DAQ list and, afterwards, assign the requested buffers.
- In order to clear a DAQ list assignment to a buffer, use FLX_ASSIGN command with XCP_PACKET_TYPE = 0x00.
- Whenever the buffer length is changed, all affected buffers (Rx or Tx, depending on direction) are being cleared of data.
- Buffers must be configured in the Xcp slave, in the A2L file and in the Fibex file. The A2L and Fibex files must be configured as specified by the ASAM specification, otherwise the master might not accept the buffers reconfiguration and throw messages like "requested bandwidth is greater than the available bandwidth for Xcp on FlexRay".

The following negative responses can be returned by FLX_ASSIGN:

- ERR_CMD_SYNTAX when FrIf_ReconfigLPdu() rejected the reconfiguration request.
- ERR_OUT_OF_RANGE when a parameter passed is incorrect (invalid Buffer ID, invalid Package Type, invalid Channel, invalid Slot ID, trying to configure a fixed parameter with a value which does not match the initial one, MAX_FLX_LEN_BUF_x is higher than MAX_FLX_LEN_BUF_x_init, MAX_FLX_LEN_BUF_x is higher than XCP_MAX_CTO/XCP_MAX_DTO + FlexRay header length)

The following negative responses can be returned by FLX_ACTIVATE/FLX_DEACTIVATE:

- ERR_OUT_OF_RANGE when a parameter passed is incorrect (invalid Buffer ID, the Buffer is an Rx type, Packet Type EV_SERV is not supported by the Buffer)

The following negative response can be returned by SET_DAQ_FLX_BUF:

- ERR_OUT_OF_RANGE when a parameter passed is incorrect (invalid buffer ID, invalid DAQ list, invalid number of buffers)

The following negative response can be returned by `GET_DAQ_FLX_BUF`:

- `ERR_OUT_OF_RANGE` when a parameter passed is incorrect (invalid DAQ list)

4.3.4.25.1. FlexRay synchronization timeout

When the `Xcp` FlexRay buffers feature is activated, while in CONNECTED mode or after a TP command has been received, a FlexRay synchronization check is done at each main function cycle execution. Until the synchronization succeeds, the `Xcp` slave does not process any command or event so it appears unresponsive to the master. The FlexRay Sync Timeout (`XcpFlexRaySyncTimeout`) parameter can be used to set a timeout value that in case of a FlexRay synchronization loss, the `Xcp` slave disconnects itself or enters a state where it is e.g. able to process incoming connections from a different bus than FlexRay.

4.3.4.26. Support for memory areas protection

With this feature you can configure which memory areas are accessible (from commands or DAQ lists) by the master. In order to enable this feature, `XcpMemoryAccessAreasSupport` must be set to true. There are two possibilities for implementation:

- Use the Xcp Memory Access Areas (`XcpMemoryAccessArea`) to configure each address range that you wish to allow access to. In case no memory area is configured, all memory access will be rejected.
- Implement a user specific callout `Xcp_ApplIsMemoryAreaAccessible()` which gives you more flexibility. Xcp Memory Access Areas Callout (`XcpMemoryAccessAreasCallout`) must be set to true and then Xcp Memory Access Areas (`XcpMemoryAccessArea`) will be disabled. Please refer to the XCP module references documentation for the prototype of the `Xcp_ApplIsMemoryAreaAccessible()` callout function.

Each entry in the Xcp Memory Access Areas (`XcpMemoryAccessArea`) has the following parameters:

- `XcpMemoryAreaStartAddress` defines the start address of this memory area. The address should be a literal address (e.g. 0xEB000).
- `XcpMemoryAreaLength` defines the length of this memory area.
- `XcpMemoryAreaAcccessType` defines the type of the access that this memory area accepts:
  - `READ_WRITE` defines that this memory area accepts all memory accesses.
  - `READ` defines that this memory area accepts only read memory accesses.
  - `WRITE` defines that this memory area accepts only write memory accesses.
- `XcpMemoryAreaAcccessScope` defines the scope of the access that this memory area accepts:
  - `ALL` defines that this memory area accepts all requests.
  - `CALIBRATION` defines that this memory area accepts only requests from commands (download/upload/modify bits/build checksum).

- `DAQ_STIM` defines that this memory area accepts only DAQ lists requests (DAQ or STIM direction).

Whenever a memory access violation is detected, the following will happen:

- If the access originated from a command (download/upload/modify bits/build checksum), error packet `ERR_ACCESS_DENIED` is thrown.
- If the access originated from a DAQ list (either direction), a `DEM` production error `XCP_E_ILLEGAL_MEMORY_ACCESS` will be thrown if `XcpIllegalMemoryAccessReportToDem` is set to `DEM` or a `DET` error (value is determined by `XcpIllegalMemoryAccessReportToDem` parameter) will be thrown if `XcpIllegalMemoryAccessReportToDem` is set to `DET`

  The `DEM` or `DET` error will be, basically, reported only for predefined DAQ lists as, for static or dynamic DAQ lists, the access to the protected area is checked at `WRITE_DAQ` command execution where the address is configured.

Note that even if a memory accesses is allowed using any of the previous methods, the access might be still checked by read/write memory callout functions (`XcpBlockWriteReadMemoryRAMMechanism`).

4.3.4.27. Support for bitwise stimulation

With this feature, you can configure `Xcp` to support bitwise stimulation within a predefined DAQ list.

In order to enable the feature, `ODT Entry Bit Offset` should be enabled and the `XcpOdtEntryBitOffset` value must be within a range from 0 to 31.

The `XcpOdtEntryBitOffset` has a dependency on the following parameters:

- `XcpAddressGranularity`, which requires the following values for the `XcpOdtEntryBitOffset`:
  - If Address Granularity is BYTE, the maximum value for this parameter is 7.
  - If Address Granularity is WORD, the maximum value for this parameter is 15.
  - If Address Granularity is DWORD, the maximum value for this parameter is 31.
- `XcpOdtEntryLength`, which should be set to 1 if `XcpOdtEntryBitOffset` is set to true

In case of a non-predefined DAQ list, the feature is supported by the WRITE_DAQ and WRITE_DAQ_MULTIPLE commands.

4.3.4.28. Support endianness for SET_MTA command

With this feature, you can configure `Xcp` to consider the platform endianness when processing the address DWORD (bytes from position 4 to 7) received within a SET_MTA command.

- When `XcpSetMtaEndianness` is enabled, byte 4 of the SET_MTA command always represents the least significant byte of the final address.

- When `XcpSetMtaEndianness` is disabled:
  - On big-endian platforms, byte 4 of the SET_MTA command represents the most significant byte of the final address.
  - On little-endian platforms, byte 4 of the SET_MTA command represents the least significant byte of the final address.

4.3.4.29. Support for user-defined callout for retrieving identification information through GET_ID type 1 command

With this feature, you can add a user-defined callout that shall retrieve the identification information when processing the GET_ID type 1 command. If the callout returns `XCP_APPL_OK`, Xcp will update the identification information as specified by ASAM, including handling the mode.

In order to enable this feature, you must enable `XcpGetIdType1Callout` parameter and set a value for the `XcpGetIdType1MaxLength` parameter. By default this parameter is disabled.

The user-defined callout function must respect the following prototype:

```
FUNC(Xcp_ApplReturnType, XCP_APPL_CODE) Xcp_ApplGetIdentification(
  uint8* IdBufferPtr,
  uint8* IdLength)
```

4.3.4.30. Support for configuring PDU attributes per connection

The following parameters can be configured for each individual connection (under `XcpConnectionCfg`):

- XcpMaxCto
- XcpMaxDto
- XcpMaxCtoPgm
- XcpMaxBS
- XcpMaxBSPgm
- XcpTimestampEnabled

The default values of connection-specific XcpMaxCto, XcpMaxDto, XcpMaxCtoPgm, XcpMaxBS, XcpMaxBSPgm parameters are taken from the global parameter with the same name (ie. the default value of the connection specific XcpMaxCto is taken from XcpGeneral/XcpMaxCto etc.).

The global counterparts of these parameters are still validated but are not used by the `Xcp` code generator.

`XcpTimestampEnabled` does not have a global counterpart and can be used to enable or disable DTO timestamps on each individual connection. Note that timestamp parameters like `XcpTimestampType` or `XcpTimestampUnit` are still global so all connections with `XcpTimestampEnabled` set to TRUE will share the values for them.

Some configuration parameters depending on the connection specific XcpMaxCto, XcpMaxDto, XcpMaxCtoPgm, XcpMaxBS, XcpMaxBSPgm might trigger false positive warnings when there is at least one connection violating the respective dependency. For instance, XcpOdtEntrySizeDaq cannot be larger than (XcpMaxDto - size of identification field). One some connections this condition might be true but on others not. In order for the integrator to be aware that using a certain value will not work in some connections (i.e. commands will be rejected, error packages will be returned, etc.), a warning will always be generated. If all connections are violating a certain restriction then an error message will be generated by `Xcp`.

4.3.4.31. RAM optimizations for large numbers of ODT entries

To enable this feature, set the configuration parameter `XcpEnableDaqOptimization` to TRUE.

By default, the parameter `XcpEnableBitOffset` is set to TRUE. However, if bit offsets are always configured to 0xFF (either in predefined DAQ lists or via WRITE_DAQ commands), then `XcpEnableBitOffset` can be set to FALSE in order to further reduce the RAM consumption.

Note that setting `XcpEnableDaqOptimization` to TRUE is not backwards compatible with regard to the RAM block referenced by parameter `XcpStoreDaqNvMBlock` (by default, NVM_BLOCK_XCP_DAQ_LISTS). This is because the optimization involves changes in the structure of the data stored in non-volatile memory, but also because explicit synchronization is used (i.e. `NvMRamBlockDataAddress` is not used anymore).

To switch to the new NvM in the safest way, proceed as follows:

1. Remove the existing NvM block.
2. Run the Service Needs Calculator.
3. Use the Memory Stack Editor in the sidebar view to finish the configuration of the block.

The `NvMNvBlockLength` parameter can be configured to a lower value than the RAM consumption is decreased. For each ODT entry, the RAM consumption is decreased by 2 bytes or 3 bytes if `XcpEnableBitOffset` is set to FALSE.

Setting `XcpEnableDaqOptimization` to FALSE is completely backwards compatible and produces the same output as before this feature. `XcpEnableBitOffset` can only be configured when `XcpEnableDaqOptimization` is TRUE and has no effect otherwise.

4.3.4.32. Support for BSW distribution

In `Xcp`, BSW distribution support is implemented based on the multiple partition concept defined by AUTOSAR.

In this type of implementation, one instance of the Xcp is assigned to each partition and the functionality is distributed across multiple partitions.

4.3.4.32.1. Functional overview

The Xcp instances are generated based on the following criteria:

- The receive, transmission, command, and event processing units are processed on each Xcp mapped Os core (one instance per Os core, for each bus type).
- A memory proxy unit is processed for each Xcp mapped Os partition (one instance per Os partition, multiple partitions per Os core can be mapped).

Calibration commands and events are processed on each partition, using the memory proxy unit.

The memory proxy unit uses the SchM sender-receiver functionality and provides a clean and consistent inter-core, inter-partition read and write access.

If the communication stack does not support BSW distribution for Xcp, several buses of the same type shall be mapped to the same Os core.

If the connection with the Xcp master is made in the context of an Os partition and Xcp receives a command from a different Os partition, the request is ignored.

The master Xcp partition needs to be configured so that the NvM is available on that partition. Also, Xcp requires to be initialized on the configured master partition.

A configurable master partition performs only the following operations:

- `Xcp_Control()`
- `Xcp_SetEvent()`
- `Xcp_SetTransmissionMode()`
- `Xcp_SetReqStoreCalReqCbk()`
- `Xcp_Init()`

If the partition of the active connection is not the same as the defined master partition when one of these operations is triggered, Xcp triggers a request to a remote dispatcher handler and data is sent using the SchM sender-receiver functionality from the master partition to the partition of the active connection to be processed, only if the operation was triggered from the master partition.

Nvm does not support BSW distribution. Only one partition has access to NvM APIs. As a consequence, the NvM must be aligned with the Xcp master partition, so NvM callbacks are processed on the same partition as the NvM is mapped to.

If the parameter XcpBswDistributionEnabled is set, the following events are not available:

- EV_RESUME_MODE
- EV_CLEAR_DAQ
- EV_STORE_DAQ

The following events are processed only on the currently active Xcp partition:

- EV_DAQ_OVERLOAD
- EV_STIM_TIMEOUT
- EV_SESSION_TERMINATED

EV_STORE_CAL uses the remote dispatcher as described above.

4.3.4.32.1.1. Asynchronous memory access areas callouts

When XcpBswDistributionEnabled is set, Xcp allows you to configure callouts for reading or writing RAM data to process calibration commands. These functions can be configured for every memory area in XcpMemoryAccessAreaReadRAMCallout and XcpMemoryAccessAreaWriteRAMCallout. If one of these parameters is not configured, then the default callouts specified by the XcpWriteMemoryRAMCallback and XcpReadMemoryRAMCallback parameters are used.

When asynchronous callouts are used, the Xcp instance that runs on the partition of the active connection dispatches the request to the memory proxy. The memory proxy calls the user callout on the partition instance that is configured for this memory area. Xcp does not send any response to the Xcp master until a Xcp_ApplWriteDataToRAMFinished or Xcp_ApplReadDataFromRAMFinished call is received.

4.3.4.32.2. Configuration options

To use the BSW distribution feature in Xcp, the XcpBswDistributionEnabled container must be enabled.

A memory proxy buffer is used to exchange data between Os partitions used by the Xcp. The size of the memory proxy buffer can be set with the XcpMemoryProxyBufferSize parameter.

In XcpMasterApplicationRef, define the Os application to which the master Xcp instance is mapped.

Map each Xcp connection and each MemoryAccessArea to an Os application by setting XcpConnectionApplicationRef and XcpMemoryAccessAreaApplicationRef.

If the access from the partition of the active connection is the same as for the MemoryAccessArea, then the access is done core-local. Otherwise, the memory proxy unit is used.

One Xcp_MainFunction for each XcpConnectionCfg is available. Their timing events must be mapped accordingly in the Rte configuration.

4.3.4.32.3. Integration notes for multi-core projects

Map each Xcp module instance in the RteBswModuleInstance list to its corresponding Os partition (indicated by their name).

For the Xcp SchM sender-receiver configuration in a BSW distribution context, consider the following hints when handling:

- Xcp events
- Xcp calibration commands
- Xcp DAQ/STIM

Xcp events

The designated master Xcp instance needs to be connected with each Xcp connection instance that is mapped to a different Os partition compared to the master Xcp instance.

Connect the ports of the above mentioned Xcp instances by adding new configuration parameters to the Rte configuration under the BSW Module Instances list, following these guidelines:

- In the master Xcp instance, under the RteBswRequiredSenderReceiverConnection list, add a new container for each Xcp connection holding a reference to an Os partition different than the master's instance (identified with ConnectionPartitionName in the example below), with the following connections:
  - In RteBswProvidedVariableDataPrototypeRef: add a reference to the NvMRequest_ConnectionPartitionName_To_Master.
  - In RteBswRequiredVariableDataPrototypeRef: add a reference to the NvMRequest_Master_From_PartitionName.
  - In RteBswProvidedDataModInstRef: add a reference to the Xcp connection instance that needs to be connected with the master Xcp instance.
- In each Xcp connection instance with a reference to an Os partition different than the master's instance (identified with ConnectionPartitionName in the example below), add a new container under the RteBswRequiredSenderReceiverConnection list, with the following connections:
  - In RteBswProvidedVariableDataPrototypeRef: add a reference to the RemoteDispatch_Master_To_ConnectionPartitionName.
  - In RteBswRequiredVariableDataPrototypeRef: add a reference to the RemoteDispatch_ConnectionPartitionName_From_Master.
  - In RteBswProvidedDataModInstRef: add a reference to the Xcp master instance.

Xcp calibration commands

Each Xcp memory access area instance needs to be connected with each Xcp connection instance holding a different reference to an Os partition and with the XcpMemoryAreaAcccessScope set to CALIBRATION or ALL.

Connect the ports of the above mentioned Xcp instances by adding new configuration parameters to the Rte configuration under the BSW Module Instances list, following these guidelines:

In the Xcp memory access area instance (identified with BSW_Xcp_1_XcpMemoryProxy_MemoryProxyPartitionName), add a new container for each Xcp connection holding a reference to an Os partition different than the memory access area's partition (identified with ConnectionPartitionName in the example below), with the following connections:

- In RteBswProvidedVariableDataPrototypeRef: add a reference to the MemoryProxyData_ConnectionPartitionName_To_MemoryProxyPartitionName.
- In RteBswRequiredVariableDataPrototypeRef: add a reference to the MemoryProxyData_MemoryProxyPartitionName_From_ConnectionPartitionName.
- In RteBswProvidedDataModInstRef: add a reference to the Xcp connection instance that needs to be connected with the Xcp memory proxy instance (identified with BSW_Xcp_1_ConnectionPartitionName).

Xcp DAQ/STIM

In addition to the previous step, for configuring the BSW_Xcp_1_XcpMemoryProxy_MemoryProxyPartitionName instances, the response instances BSW_Xcp_1_XcpMemoryProxyResponse_ConnectionPartitionName need to be connected with the Xcp connection instances. The MemoryProxyResponse instances are generated if there is at least one entry in the Xcp Memory Access Area list with the XcpMemoryAreaAccessScope set to DAQ_STIM or ALL.

Connect the ports of the above mentioned Xcp instances by adding new configuration parameters to the Rte configuration under the BSW Module Instances list, following these guidelines:

In the Xcp memory access area instance (identified with BSW_Xcp_1_XcpMemoryProxyResponse_ConnectionPartitionName), add a new container for each Xcp memory access area partition (identified with MemoryProxyPartitionName in the example below) holding a reference to an Os partition different than the Xcp connection's partition, with the following connections:

- In RteBswProvidedVariableDataPrototypeRef: add a reference to the EventId_MemoryProxyPartitionName_To_ConnectionPartitionName.
- In RteBswRequiredVariableDataPrototypeRef: add a reference to the EventId_ConnectionPartitionName_From_MemoryProxyPartitionName.
- In RteBswProvidedDataModInstRef: add a reference to the Xcp connection instance that needs to be connected with the Xcp memory proxy instance (identified with BSW_Xcp_1_XcpMemoryProxy_MemoryProxyPartitionName).

4.3.4.32.3.1. Configuring Xcp BSW distribution with Rte Editor

If you use the Rte Editor for configuring the Xcp-related BSW distribution artifacts, consider the following hints.

Configuring Xcp BSW distribution with Rte Editor

Step 1
Open the Rte Editor and go to the **Partitioning** tab.

Step 2
Map each BSW_Xcp_1_<partition> instance to the partition indicated in the name. The partition names are used as suffixes.

In case of artifacts where the suffix is <partition 1>_From_<partition 2>, map this instance to the <partition 1> partition.

Step 3
Go to the **Event Mapping** tab.

Step 4
Map the following events to the appropriate partitions:

Event	**Partition**
DataReceivedEvent_Xcp_MemoryProxyHandler_<partition 1>_From_<partition 2>	<partition 1>
DataReceivedEvent_Xcp_MemoryProxyResponseHandler_<partition 1>_From_<partition 2>	<partition 1>
DRE_RemoteDispatch_<partition>_From_Master	<partition>

Step 5
Go to the **BSW Required SR Connections** tab.

You need to connect the provided variable data prototypes to required variable data prototypes. In the following instructions, the partition referenced by the configuration parameter `XcpMasterApplicationRef` is indicated by <master>. To do the mapping, expand both instances in the left and right panels, select the desired variable data prototypes and then press the button **Create a required BSW sender receiver connection with the selected data prototypes**.

Step 5.1
For each BSW_Xcp_1_<partition> where <partition> is not the master application (`XcpMasterApplicationRef`), map the RemoteDispatch_Master_To_<partition> on the left to RemoteDispatch_<partition>_From_Master on the right.

Step 5.2
In order to map the variable data prototypes under each BSW_Xcp_1_XcpMemoryProxy_<partition> instance, first identify all partitions that the Xcp connections are mapped to via the `XcpConnectionApplicationRef` parameter.

Based on your project requirements, now identify the partitions where the data is accessed when Xcp is connected to each connection configured in `XcpConnectionCfg`.

At the end you have a map consisting of pairs <partition 1> - <partition 2> with the following semantics: When the connection is established on <partition 1>, then it accesses data located on <partition 2>.

With this in mind, locate the BSW_Xcp_1_XcpMemoryProxy_<partition 1> instance on the right panel and select the MemoryProxyData_<partition 1>_From_<partition 2>.

On the left panel, locate the BSW_Xcp_1_<partition 2> instance and select the MemoryProxyData_<partition 2>_To_<partition 1> variable data prototype.

Connect the above variable data prototypes. Note that on the left panel, you have as many BSW_Xcp_1_<partition> instances as the number of unique partitions that you have mapped to Xcp connections. On the right panel you have as many BSW_Xcp_1_XcpMemoryProxy_<partition> instances as the number of unique partitions that you have mapped to Xcp Memory Areas.

Step 5.3
For each BSW_Xcp_1_XcpMemoryProxyResponse_<partition 1>_From_<partition 2> instance on the right panel, select the EventId_<partition 1>_From_<partition 2>.

On the left panel, search for BSW_Xcp_1_XcpMemoryProxy_<partition 2> and select the EventId_<partition 2>_To_<partition 1>.

Connect the selected variable data prototypes.

Step 6
Run the **Calculate Service Needs** assistant.

4.4. XcpR module user guide

4.4.1. Overview

This user guide describes the `Xcp Router` module. From this user guide you learn about the basic functionality of the `Xcp Router`. You also learn which related modules are necessary to configure the `Xcp Router` module. The `Xcp Router` module reference provides further information on how to configure the `Xcp Router` itself.

Note that this user guide is intended for readers who have good knowledge of AUTOSAR and about the purpose of the `Xcp` and `XcpR`. The information provided here helps you to integrate the `XcpR` in your AUTOSAR project.

- Section 4.4.2, “Background information” provides an overview of the basic functionality of the `XcpR`.
- Section 4.4.3, “Configuring the XcpR module” provides information on related modules that are needed in order to configure the `XcpR`.
- Section 4.4.4, “XcpR integration notes” provides information on how to integrate the `XcpR` into your project.

For more details on how to configure the `XcpR` itself, see the parameter descriptions provided in the `XcpR` module reference Chapter 5, “ACG8 XCP Stack module references“.

4.4.2. Background information

XcpR is a module used for routing XCP messages to an XCP instance located on the same ECU (local Xcp slave) or on a remote ECU (remote Xcp slave). The separation between the protocol layer and the transport layer means that XcpR is designed to be used over several network types including CAN, CAN-FD and FlexRay. XcpR is implemented based on internal requirements. Figure 4.3, "XcpR in the XCP Stack" shows the location of the XcpR in the context of the XCP Stack.

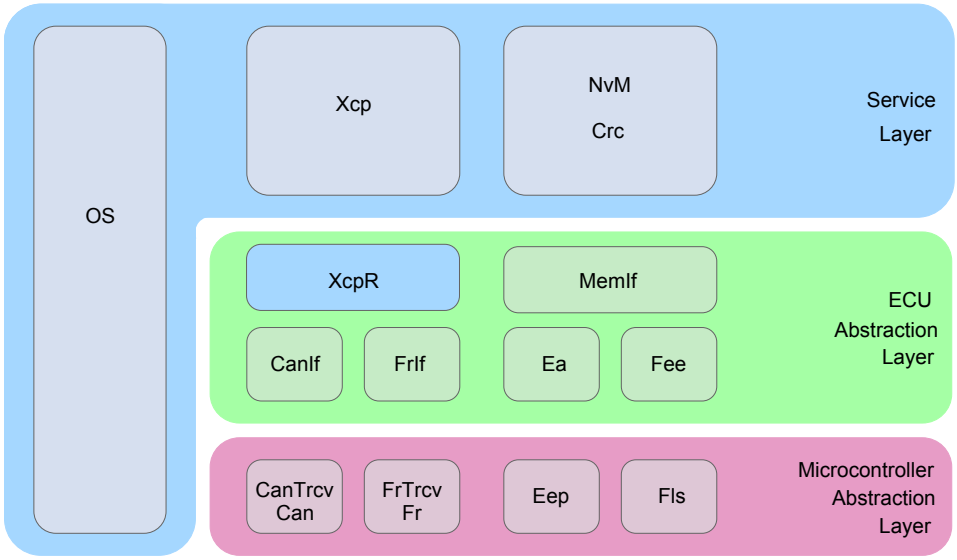

Figure 4.3. XcpR in the XCP Stack

4.4.2.1. Functional overview

XcpR is used when the Xcp master and the Xcp slave are not connected to the same bus. For example, the Xcp master is connected to an external bus but the Xcp slave is connected only to an internal bus. The XcpR receives the Xcp packages on the external bus and sends them to the Xcp slave on the internal bus. XcpR supports CAN, CAN-FD and FlexRay as bus types.

XcpR offers also the possibility to communicate between an Xcp master with two different Xcp slaves located on different ECUs. For example, one Xcp slave is located on an ECU connected to the same bus as the Xcp master and the second Xcp slave is located on a second ECU connected to an internal bus. The XcpR receives the Xcp packages and, depending on the value of the Mode parameter received in a connect command, the XcpR routes the Xcp messages either to the first or to the second slave instance.

When two Xcp slave instances are configured on different ECUs, the interactions and functionality of the XcpR module are as depicted in the following diagram:

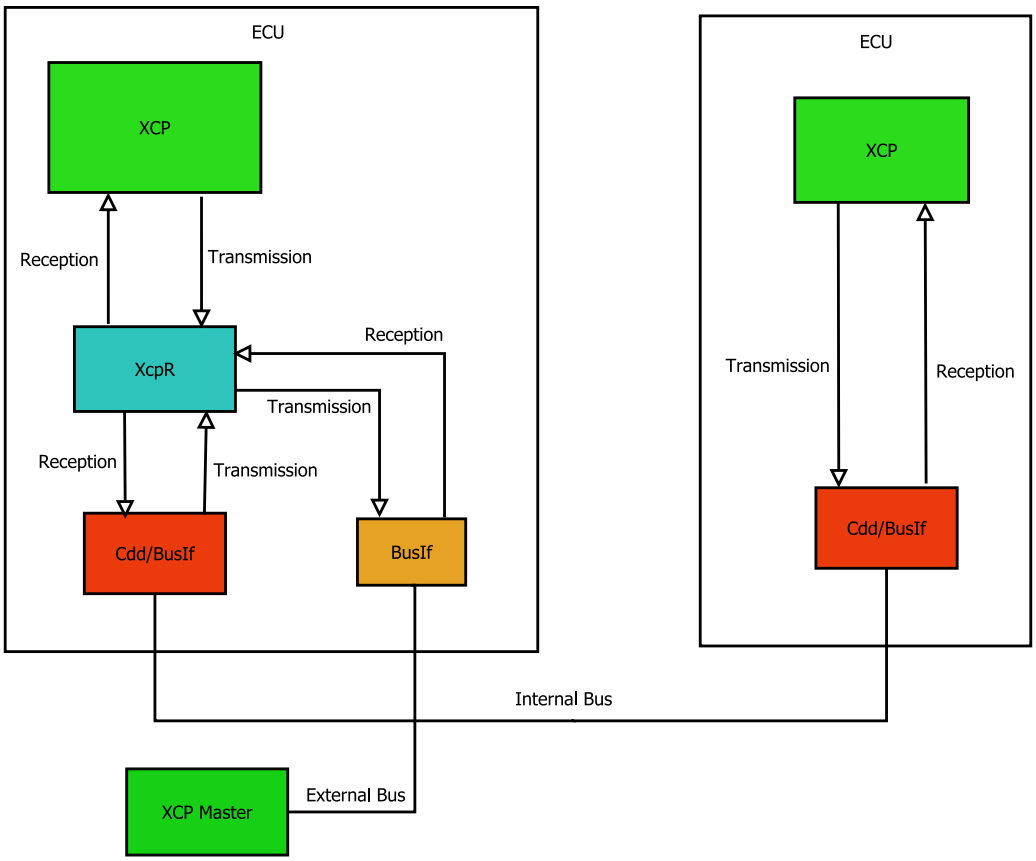

Figure 4.4. `XcpR` interactions and functionality

The connection between `XcpR` and the local `Xcp` slave is always configured over a CDD. For the connection with the remote `Xcp` slave, it is possible to configure a CAN, CAN-FD, FlexRay, or CDD connection.

4.4.3. Configuring the XcpR module

4.4.3.1. Configuring routing paths

Each routing path is composed of a source and a destination. To configure a source, it is necessary to select the configuration `XcpRSourceInterfaceType`. If the source of the messages is from a master `Xcp`, the parameter `XcpRIsSrcMasterConnection` must be enabled. The parameter `XcpRPdu` is used to configure the type of the PDU (Tx or Rx). For Rx sources, the parameter `XcpRRxSourcePduReference` must contain a valid reference to an `EcuC` Rx PDU. `XcpRRxSourcePduId` is configured with a zero-based consecutive value.

To configure a destination, it is necessary to select the configuration `XcpRDestinationInterfaceType`. If the destination is linked to the `Xcp` master, the parameter `XcpRIsDestMasterConnection` must be enabled. The `XcpRPdu` parameter is used to select if the destination is either a transmission (Tx) PDU or a reception (Rx)

PDU. If an Rx type is configured, then XcpRRxDestinationPduReference should contain a valid reference to an EcuC Rx PDU. In the upper layer configuration container, the following parameters can be configured:

- the short name of the upper layer (XcpRUpperLayerModuleShortName parameter)
- the header file provided by the upper layer (XcpRUpperLayerHeaderFile parameter)
- the API that is used for RxIndication on the upper layer (XcpRRxIndicationFunctionName parameter)
- the API used for TxConfirmation (XcpRTxConfirmationFunctionName parameter)
- the trigger transmit function (XcpRTriggerTransmitFunctionName parameter)

For a Tx destination, the parameter XcpRTxDestinationPduReference should contain a valid reference to an EcuC Tx PDU. The parameter XcpRLowerLayerDestinationPduId must be configured with the ID of the PDU that is used for the outgoing lower layer transmit. XcpRTxDestinationPduId must be configured with the PDU identifier that is used by the lower layer BSW module for XcpR_TxConfirmation() or XcpR_TriggerTransmit(). This ID must be zero-based and consecutive.

In the XcpRRoutingPaths tab, you configure a routing path. For each entry, a source (XcpRSourcePduRef parameter) is linked to a destination (XcpRDestinationPduRef parameter). If the XcpRMultipleSlavesEnabled parameter is set to true, then two routing paths can be configured for one source.

NOTE **Two routing paths must be configured**

For configuring communication paths between an Xcp master and an Xcp slave, it is required to configure two routing paths: one from master to slave, and one from slave to master.

4.4.3.2. Configuring connection groups

To enable the connection group functionality, configure an entry in XcpRRoutingToConnectionMapping. For every connection group entry, it is necessary to configure at least a routing path for the request from the Xcp master to the Xcp slave, and one path for routing the responses from the Xcp slave to the Xcp master. In XcpRRoutingToConnectionMapping, XcpRRoutingMapping container, use the parameter XcpRRoutingPathRef to reference a configured routing path.

4.4.3.3. Configuring general parameters

The parameter XcpRQueueSize is used for configuring the queue size of the XcpR internal ring-buffer. For every message received on XcpR, one entry is saved in the buffer containing the data of the message and a PduLength (sizeof PduLengthType), a PduId (sizeof PduIdType), and the PduType (one byte). If the XcpR receives messages for which there is not sufficient space in the queue, it discards these messages and stores a Det error.

NOTE **The XcpR queue size affects the resource consumption**

The parameter `XcpRQueueSize` might have a major impact on the resource consumption of your project.

4.4.4. XcpR integration notes

The integration notes are intended to provide a guide for integrating the `XcpR` module into your project. The points addressed are specific to issues that may arise due to limitations in the `XcpR` or in the related modules mentioned in Section 4.4.3, “Configuring the XcpR module”.

4.4.4.1. Support for multiple routing paths

`XcpR` supports the routing of received XCP messages to different destinations. So it is possible to configure multiple sources and destinations for XCP messages. For every source it is possible to configure one or two destinations. If the source is configured as a source from the `Xcp` master, it is possible to link this source to two destinations: one to the local `Xcp` slave and one to the remote `Xcp` slave. The destination to the local `Xcp` slave is always configured as CDD. For the destination to the remote `Xcp` slave, there is no limitation regarding the type of transmission layer.

4.4.4.2. Configuring PDU attributes

For every configured source and destination, PDU attributes are configurable.

For every source, the following attributes are configurable:

- the maximum length of a CTO packet `XcpRSrcMaxCto`
- the maximum length of a DTO packet `XcpRSrcMaxDto`
- the timestamp `XcpRSrcTimestampType`
- optionally, the maximum length of command transfer objects `XcpRSrcMaxCtoPgm`

For every destination, the following attributes are configurable:

- the maximum length of a CTO packet `XcpRDestMaxCto`
- the maximum length of a DTO packet `XcpRDestMaxDto`
- the timestamp `XcpRDestTimestampType`
- optionally, the maximum length of command transfer objects `XcpRDestMaxCtoPgm`

NOTE **Same PDU attribute values for source and destination**

For every routing path, the PDU attributes of a source and a destination should be configured with the same values as the ones configured in the Xcp slave for which the routing path was configured.

4.4.4.3. Support for routing messages between an Xcp master and an Xcp slave

XcpR can route XCP messages to an XCP instance located on the same ECU or on a remote ECU. If for one source there are two possible destinations configured, then XcpR establishes the active routing path based on the mode sent in the connect command from the Xcp master. If the command has mode 0, it is considered as a request for connecting to the local Xcp slave, and if the command has mode 1, it is considered as a request for connecting to the remote Xcp slave. XcpR forwards the connect request to the Xcp slave, and forwards the received response to the Xcp master. If the response is positive, and XcpR receives a TxConfirmation from the external bus, then it sets its status to connected, and all the following messages are routed between the Xcp master and the Xcp slave for which the connection was established.

For sending messages to the other Xcp slave instance, it is necessary for the Xcp master to send a disconnect command. After the positive response for this request is forwarded to the Xcp master and a TxConfirmation is received from the external bus, XcpR is set to a disconnected state, and is once again available for receiving connect requests.

NOTE **XcpR connection state and resulting behavior**

If XcpR is not connected, then all messages except the connect command are discarded without forwarding them to an Xcp slave.

If XcpR is connected to an Xcp slave instance, it forwards all messages from the Xcp master without interpreting the data in the message, except for the disconnect command.

When XcpR transmits a message from the local Xcp slave to the Xcp master, and it receives a TxConfirmation for this transmission or a trigger transmit request, it forwards it to the Xcp slave.

4.4.4.4. Support for transmission and reception of XCP packages from ISR context

4.4.4.4.1. Message transmission from XcpR to Xcp master and Xcp slave

There are four possibilities to trigger the transmission of messages from the XcpR to the Xcp master and Xcp slave:

- From the context of XcpR_MainFunction. In this case, as long as there is data to be transmitted, messages are packed and sent on the bus based on the main processing cycle of the module.

- From the context of `XcpR_TxConfirmation()`, if configured with configuration parameter `XcpRDestPduSupportTxFromTxConfirmation`. In this case, data is packed and transmitted from the TxConfirmation of the underlying layer, from `XcpR_MainFunction`.
- If reception from RxIndication (`XcpRSrcPduSupportRxFromRxIndication`) is enabled, it is possible to enable the transmission from the context of receiving a message from `RxIndication()`. This is possible if the parameter `XcpRRxSrcPduSupportTxFromRxIndication` is enabled.
- If reception from XcpR_Transmit (`XcpRSrcPduSupportRxFromXcpRTransmit`) is enabled, it is possible to enable the transmission from the context of receiving a message from `XcpR_Transmit()`. This is possible if the parameter `XcpRRxSrcPduSupportTxFromRxIndication` is enabled.

NOTE **Impact of message transmission on the CPU load**

It is not the job of the `XcpR` module to assure that, based on the configuration, the transmission of messages does not create an overhead in the CPU load. As long as the transmission is done from several contexts and especially from an interrupt context, during integration phase it shall be assured that the time needed for packing and sending messages on the bus does not affect the runtime of the application and does not increase the CPU load.

4.4.4.4.2. Message reception from Xcp master or Xcp slave

There are two possibilities to process the incoming messages from `Xcp` master or slave to `XcpR`:

- From the context of `XcpR_MainFunction`. In this case, as long as there is data to be received, messages are received on the bus and unpacked based on the main processing cycle of the module.
- From the context of `XcpR_RxIndication()`, if configured with configuration parameter `XcpRPduSupportRxFromRxIndication`. In this case, data is unpacked and processed from the `RxIndication` of the underlying layer. In the case of connection to a local `Xcp` slave, the messages are received via `XcpR_Transmit`, if configuration parameter `XcpRSrcPduSupportRxFromXcpRTransmit` is enabled.

NOTE **Impact of message reception on the CPU load**

It is not the job of the `XcpR` module to assure that, based on the configuration, the reception of messages does not create overhead in the CPU load. As long as the reception is done from several contexts, especially from an interrupt context, during integration phase it shall be assured that the time needed for receiving and unpacking messages on the bus does not affect the runtime of the application and does not increase the CPU load.

4.4.4.5. Transmission failure monitoring

Each message accepted by the lower layer for transmission is monitored for a period of time (defined by parameter `XcpRTxBusTimeout`) and, during that monitoring time, `XcpR` expects the message to be transmitted successfully, i.e. the `TxConfirmation` is received.

In order to retry the whole message transmission, you must configure XcpRTxBusRetry. If the message times out and the number of configured retries is not reached, the XcpR retries the message transmission process. That means XcpR calls xxxIf_Transmit() once again with the same message. If XcpRTxBusRetry is configured to 255, XcpR retries indefinitely to transmit the package.

If the message times out for the number of configured retries, the connection is deemed compromised and the message is discarded. If the bus on which XcpR did not receive TxConfirmation is the external bus, the XcpR autonomously disconnects from the Xcp master and triggers a disconnect request to the Xcp slave.

If you have configured event packets (XcpREventPacketEnabled), then before disconnecting, XcpR tries to notify the master with the event packet EV_SESSION_TERMINATED.

4.4.4.6. Support for transmission retry

Each message for which the transmission API returns the E_NOT_OK value triggers a retry for transmission. If the retransmission is not successful for the configured XcpRTxRetryCount number of times, XcpR discards the message. If the transmission failed on the external bus (messages for Xcp master), XcpR triggers an autonomous disconnect. That means XcpR sends a disconnect command to the Xcp slave so that both XcpR and the Xcp slave remain in the same connection state. If you have configured event packets (XcpREventPacketEnabled), the before disconnecting, XcpR tries to notify the master with the event packet EV_SESSION_TERMINATED.

4.4.4.7. Support connections on CAN-FD

The maximum data length of a CAN message and therefore maximum length of an XcpR on CAN message is MAX_DLC = 8. The maximum data length of an XcpR on CAN-FD message equals one of the following values, i.e. MAX_DLC (XcpRMaxDlc) = 8, 12, 20, 16, 24, 32, 48, 64.

For CAN-FD, the data length of the XcpR packages is determined in two ways:

- If XcpRDestinationCanFdMaxDlcRequired (MAX_DLC_REQUIRED) is set, the length of packages to be sent on the bus is given by the configuration of parameter XcpRDestinationCanFdMaxDlc. The size of the packages that are received on the bus is given by the configuration parameter XcpRSourceCanFdMaxDlc, when XcpRSourceCanFdMaxDlcRequired is enabled.
- If XcpRDestinationCanFdMaxDlcRequired (MAX_DLC_REQUIRED) is not set, the length of the packages equals the next higher valid values of CAN-FD DLC. If XcpRSourceCanFdMaxDlcRequired is not set, the length of the packages is calculated as for the destinations.

You can define the value to be used as *fill byte* for the transmitted packages by configuring the parameter XcpRDestinationCanFdFillValue. For the received packages, there is no way to distinguish which are the fill bytes. So no configuration parameter is available for setting a value for the byte.

4.4.4.8. Support for connections on CDD

The maximum data length of a CDD message and therefore maximum length of a `XcpR` on CDD message is determined by the bus type represented by the complex device driver.

For a CDD destination to a local `Xcp` slave (Rx destination), it is necessary to configure the following:

- name of the upper layer (`XcpRUpperLayerModuleShortName`)
- the header file (`XcpRUpperLayerHeaderFile`)
- the parameter with the name of the header that contains the transmit function name (configured in the `XcpCddTransmitFunctionName` parameter)
- the Rx Indication API (configured in the `XcpRRxIndicationFunctionName` parameter)
- the Tx Confirmation API (configured in `Xcp_CddTxConfirmation`)
- the TriggerTransmit API (configured in `Xcp_CddTriggerTransmit` parameter)

For a CDD destination used for bus (Tx destination), it is necessary to configure:

- the short name (`XcpRCddShortName`)
- the header file (`XcpRCddHeaderFile`)
- the transmission function (`XcpRCddTransmitFunctionName`)

4.4.4.9. Support for connections on FlexRay

The maximum data length of a FlexRay message and therefore maximum length of an `XcpR` on FlexRay message is 254 bytes. This can be configured per source and destination in the parameter `XcpRSourceMaxFlexMsgLength` respectively `XcpRDestinationMaxFlexMsgLength`.

For every FlexRay connection, it is necessary to configure the NAX (`XcpRSourceFlxNodeAddress` or `XcpRDestinationFlxNodeAddress`) and the header alignment (`XcpRSourceFlxHeaderAlignment` or `XcpRDestinationFlxHeaderAlignment`).

4.4.4.10. Support for XcpR on FlexRay sequence correction

In order to enable FlexRay sequence correction, you must enable the `XcpRSourceSequenceCorrectionEnabled` and `XcpRDestinationSequenceCorrectionEnabled` parameters for the source and destination of a routing path.

Sequence correction consists of a counter (1 byte) that is sent with each frame. The counter starts with 0, and it is incremented after each successfully transmitted packet.

4.4.4.11. Support for packing multiple PDUs in one frame

XcpR supports the packing of multiple PDUs in one frame.

The number of packages packed into a frame is variable and depends on how many packages are available for transmission/reception and the maximum size of the frame. All kinds of packages are packed into the frame, meaning that the Xcp master should be able to unpack both CTOs and DTOs from each one of the configured transmission PDUs. The packing is done as described by the ASAM.

If packing is enabled for a source, XcpR unpacks the packages from the received frame. Based on the type of the selected destination, it either transmits a frame or sends individual packets.

This feature is supported only for FlexRay and Ethernet. It can be individually activated for some of the configured connections that support the feature.

For the CAN protocol or connections over CDD, this feature is not supported.

4.4.4.12. Support for connections on Ethernet

If the desired protocol is TCP/IP, configure for each XcpR connection a single socket connection in the SoAd module. This socket connection must contain mappings for the PDUs configured for the XcpR's Ethernet configured source and destination.

This socket connection must be referenced by the path XcpRDestinationConnectionSoAdConfigRef/SoAdPduRouteDest/SoAdTxSocketConnOrSocketConnBundleRef. The SoAdTxSocketConnOrSocketConnBundleRef must reference a socket connection, not a socket connection group.

An XcpR Ethernet connection is defined by a connection to either the master Xcp or a remote slave Xcp. A connection between XcpR and the master Xcp is defined in the XcpR configuration by an XcpR source from the master Xcp and an XcpR destination to the master Xcp.

If the UDP/IP protocol is used for the selected XcpR connection, connections with two sockets must be provided: one for the XcpR Ethernet configured source and another one for the XcpR Ethernet configured destination. These connections with two sockets must belong to the same socket connection group.

The socket configured for the XcpR destination must be referenced by the path XcpRDestinationConnectionSoAdConfigRef/SoAdPduRouteDest/SoAdTxSocketConnOrSocketConnBundleRef. The SoAdTxSocketConnOrSocketConnBundleRef must reference a socket connection, not a socket connection group.

For each available XcpR over Ethernet connection, the XcpR may open the necessary sockets in its initialization routine. However, if you configure the XcpRSourceAutoOpenSoCon or XcpRDestinationAutoOpenSoCon parameter to false, the necessary sockets are opened by the SoAd module. If XcpRSourceAutoOpenSoCon or XcpRDestinationAutoOpenSoCon is set to true, make sure that XcpR_Init() is called after the SoAd_Init() in the initialization routine.

It is not allowed to share the same socket connection group between XcpR Ethernet connections.

If the UDP/IP protocol is used for the selected XcpR connection, SoAdSocketAutomaticSoConSetup must be set to false, and XcpRDestinationAutoOpenSoCon must be set to true.

NOTE **Bidirectional and independent sockets**

For XcpR Ethernet connections on the internal bus, the same address and port are always used, regardless if TCP/IP or UDP/IP connections are configured.

For connections between XcpR and the master Xcp, the ASAM specification is implemented, which can manage an independent socket for an Rx PDU and another socket for a Tx PDU.

4.4.4.13. Support for timestamp

XcpR supports the same types of timestamps as Xcp. The timestamp is configured per source and destination (XcpRSrcTimestampType and XcpRDestTimestampType parameters):

- no timestamp (NO_TIME_STAMP)
- timestamp on 1 byte (ONE_BYTE)
- timestamp on 2 bytes (TWO_BYTE)
- timestamp on 4 bytes (FOUR_BYTE)

NOTE **Timestamp alignment**

XcpR does not modify the data of a packet. So it is necessary to configure the same timestamp type for both the source and destination of a routing path. This timestamp type should always be aligned with the type configured in the Xcp slave.

4.4.4.14. Support for connection groups

XcpR supports the configuration of connection groups. Connection groups are lists of logically grouped routing paths. A connection group contains at least a routing path from the Xcp master to the Xcp slave and one path from Xcp slave to Xcp master. In addition to these routing paths, it is possible to add paths for DAQ or STIM packages.

XcpR does not modify the data of a packet. So it is necessary to configure the same timestamp type for both the source and destination of a routing path. This timestamp type should always be aligned with the type configured in the Xcp slave.

When connection group support feature is enabled, and a message is sent to XcpR, it is checked if the source PDU ID is assigned to the active connection group. This is to verify that the connection was established on a

routing path configured for this connection group. If this condition is true, `XcpR` forwards the message on the routing path configured for the source PDU ID. If a message is received for a PDU ID that is not member of the active connection group, this message is discarded.

5. ACG8 XCP Stack module references

5.1. Overview

This chapter provides module references for the ACG8 XCP Stack product modules. These include a detailed description of all configuration parameters. Furthermore this chapter lists the application programming interface with all data types, constants and functions.

The content of the sections is sorted alphabetically according the EB tresos AutoCore Generic module names.

For further information on the functional behavior of these modules, refer to the chapter ACG8 XCP Stack user's guide.

5.1.1. Notation in EB module references

EB notation may differ from the AUTOSAR standard notation in the software specification documents (SWS). This section describes the notation of *default value* and *range* fields in the EB module references.

5.1.1.1. Default value of configuration parameters

If there is no default value specified for a parameter, the default value field is omitted to prevent ambiguity with parameters that have `--` as default values.

Example: The parameter `BswMCompuConstText` of the `BswM` module of EB tresos AutoCore Generic 8 Mode Management has no default value field, therefore it is omitted.

5.1.1.2. Range information of configuration parameters

The range of a configuration parameter contains an upper and a lower boundary. However, in special cases the range of allowed values can be computed by means of an XPath function that is evaluated at configuration time. An XPath function can either be a standard `xpath:<function>()` or a custom `cxpath:<function>()` function. The range of a configuration parameter may be computed based on other configuration parameters that are referenced from the XPath function. For more information on custom XPath functions, see section *Custom XPath Functions API* of the EB tresos Studio developer's guide.

Example: The parameter `BswMCompuConstText` of the `BswM` module of EB tresos AutoCore Generic 8 Mode Management has the custom XPath function `cxpath:getCompuMethodsVT()` in the range field which provides the allowed values.

5.2. Xcp

5.2.1. Configuration parameters

Containers included		
Container name	**Multiplicity**	**Description**
CommonPublishedInformation	1..1	**Label:** Common Published Information Common container, aggregated by all modules. It contains published information about vendor and versions.
PublishedInformation	1..1	**Label:** EB Published Information Additional published parameters not covered by CommonPublishedInformation container.
XcpDefensiveProgramming	1..1	**Label:** Defensive Programming Options Parameters for defensive programming
XcpConfig	1..1	This container contains the configuration parameters and sub-containers of the Xcp module supporting multiple configuration sets. This container is a MultipleConfigurationContainer, i.e. this container and its sub-containers exist once per configuration set.
RamOptimizations	1..1	**Label:** RAM Optimizations
ReportToDem	1..1	**Label:** Production error handling Production error handling
XcpBswDistribution	0..1	**Label:** Xcp Bsw Distribution Defines the configuration parameters related to the Bsw distribution functionality.
XcpGeneral	1..1	Defines the general configuration parameters of the Xcp.
XcpUserCommand	0..n	This container contains the configuration parameters of Xcp user commands. User commands must not be used to implement functionalities done by other services.

Parameters included	
Parameter name	**Multiplicity**
IMPLEMENTATION_CONFIG_VARIANT	1..1

Parameter Name	IMPLEMENTATION_CONFIG_VARIANT	
Label	Config Variant	
Multiplicity	1..1	
Type	ENUMERATION	
Default value	VariantPreCompile	
Range	VariantPreCompile	
Configuration class	**VariantPreCompile:**	VariantPreCompile

5.2.1.1. CommonPublishedInformation

Parameters included	
Parameter name	**Multiplicity**
ArMajorVersion	1..1
ArMinorVersion	1..1
ArPatchVersion	1..1
SwMajorVersion	1..1
SwMinorVersion	1..1
SwPatchVersion	1..1
ModuleId	1..1
VendorId	1..1
Release	1..1

Parameter Name	ArMajorVersion	
Label	AUTOSAR Major Version	
Description	Major version number of AUTOSAR specification on which the appropriate implementation is based on.	
Multiplicity	1..1	
Type	INTEGER_LABEL	
Default value	2	
Configuration class	**PublishedInformation:**	

Origin	Elektrobit Automotive GmbH

Parameter Name	**ArMinorVersion**
Label	AUTOSAR Minor Version
Description	Minor version number of AUTOSAR specification on which the appropriate implementation is based on.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	0
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**ArPatchVersion**
Label	AUTOSAR Patch Version
Description	Patch level version number of AUTOSAR specification on which the appropriate implementation is based on.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	0
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**SwMajorVersion**
Label	Software Major Version
Description	Major version number of the vendor specific implementation of the module.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	2
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**SwMinorVersion**
Label	Software Minor Version
Description	Minor version number of the vendor specific implementation of the module. The numbering is vendor specific.

Multiplicity	1..1
Type	INTEGER_LABEL
Default value	12
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**SwPatchVersion**
Label	Software Patch Version
Description	Patch level version number of the vendor specific implementation of the module. The numbering is vendor specific.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	2
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**ModuleId**
Label	Numeric Module ID
Description	Module ID of this module from Module List
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	212
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**VendorId**
Label	Vendor ID
Description	Vendor ID of the dedicated implementation of this module according to the AUTOSAR vendor list
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	1
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	Release
Label	Release Information
Multiplicity	1..1
Type	STRING_LABEL
Default value	
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

5.2.1.2. PublishedInformation

Parameters included	
Parameter name	**Multiplicity**
PbcfgMSupport	1..1

Parameter Name	PbcfgMSupport
Label	PbcfgM support
Description	Specifies whether or not the Xcp can use the PbcfgM module for post-build support.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

5.2.1.3. XcpDefensiveProgramming

Parameters included	
Parameter name	**Multiplicity**
XcpDefProgEnabled	1..1
XcpPrecondAssertEnabled	1..1
XcpPostcondAssertEnabled	1..1
XcpStaticAssertEnabled	1..1
XcpUnreachAssertEnabled	1..1

Parameters included	
XcpInvariantAssertEnabled	1..1

Parameter Name	XcpDefProgEnabled	
Label	Enable Defensive Programming	
Description	Enables or disables the defensive programming feature for the module Xcp. Note: This feature is dependent on the use of the development error detection module. To use the defensive programming feature, proceed as follows: 1. Enable development error detection 2. Enable defensive programming 3. Enable assertions as required	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	XcpPrecondAssertEnabled	
Label	Enable Precondition Assertions	
Description	Enables handling of precondition assertion checks reported from the module Xcp. Dependency on parameter(s): ► Enable Development Error Detection (`XcpDevErrorDetect`): must be enabled ► Enable Defensive Programming (`XcpDefProgEnabled`): must be enabled	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	XcpPostcondAssertEnabled
Label	Enable Postcondition Assertions

Description	Enables handling of postcondition assertion checks reported from the module Xcp. Dependency on parameter(s): ► Enable Development Error Detection (`XcpDevErrorDetect`): must be enabled ► Enable Defensive Programming (`XcpDefProgEnabled`): must be enabled
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpStaticAssertEnabled**
Label	Enable Static Assertions
Description	Enables handling of static assertion checks reported from the module Xcp. Dependency on parameter(s): ► Enable Development Error Detection (`XcpDevErrorDetect`): must be enabled ► Enable Defensive Programming (`XcpDefProgEnabled`): must be enabled
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpUnreachAssertEnabled**
Label	Enable Unreachable Code Assertions
Description	Enables handling of unreachable code assertion checks reported from the module Xcp. Dependency on parameter(s): ► Enable Development Error Detection (`XcpDevErrorDetect`): must be enabled ► Enable Defensive Programming (`XcpDefProgEnabled`): must be enabled

Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**XcpInvariantAssertEnabled**
Label	Enable Invariant Assertions
Description	Enables handling of invariant assertion checks reported from functions of the module Xcp. Dependency on parameter(s): ► Enable Development Error Detection (`XcpDevErrorDetect`): must be enabled ► Enable Defensive Programming (`XcpDefProgEnabled`): must be enabled
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

5.2.1.4. XcpConfig

Containers included		
Container name	**Multiplicity**	**Description**
XcpDaqList	0..n	Defines `PREDEFINED` and `STATIC` configurable data acquisition (DAQ) lists. To configure DAQ lists, proceed as follows: ► `PREDEFINED` DAQ lists (i.e. lists with `XcpDaqListNumber` less than `XcpMinDaq`): Configure the general parameters in the container `XcpDaqList` including all object descriptor tables (ODTs, sub-containers `XcpOdt`) and their parameters. ► `STATIC` DAQ lists (i.e. lists with `XcpDaqListNumber` >= `XcpMinDaq` if the parameter `XcpDaqConfigType`

Containers included		
		is set to DAQ_STATIC): Configure the general parameters in the container XcpDaqList. Do not configure any XcpOdt containers. Note: DYNAMIC DAQ lists (i.e. lists with XcpDaqListNumber >= XcpMinDaq if the parameter XcpDaqConfigType is set to DAQ_DYNAMIC) are not configured in this container. Dependency on parameter(s): ► DAQ list support (XcpDaqSupported): must be enabled to configure any DAQ lists **Optimization Effect:** ► **ROM reduction (config):** Configuring less DAQ lists reduces the ROM consumption of the module configuration. ► **RAM reduction (config):** Configuring less DAQ lists reduces the RAM consumption of the module configuration. ► **Execution time reduction (code):** Configuring less DAQ lists reduces the execution time of the module code.
XcpDemEventParameterRefs	1..1	**Label:** Dem Events References Reference to the DemEventParameter elements that shall be invoked using the API Dem_ReportErrorStatus() if 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	0..n	Defines the configuration of event channels on the Xcp slave. Dependency on parameter(s): ► DAQ list support (XcpDaqSupported): must be enabled to configure any event channel **Optimization Effect:**

Containers included		
		▶ **RAM reduction (config):** Configuring less event channels reduces the RAM consumption of the module configuration. ▶ **ROM reduction (config):** Configuring less event channels reduces the ROM consumption of the module configuration. ▶ **Execution time reduction (code):** Configuring less event channels reduces the execution time of the module code.
XcpPdu	1..n	Defines PDU information. A PDU may be either a transmission (Tx) PDU or a reception (Rx) PDU.
XcpConnectionCfg	1..n	Groups PDU information related to a connection via a specified bus (Can/CanFD/FlexRay/Ethernet). Multiple connections via the same bus type can be configured. Dependency on parameter: ▶ If the connection type is Xcp over Ethernet (`XcpEthernetConnectionType`), the following conditions must be met: ▶ The socket connections (`SoAdSocketConnection`) configured for the PDUs belonging to the selected connection must belong to the same socket connection group (`SoAdSocketConnectionGroup`). ▶ Usually, only one socket connection (`SoAdSocketConnection`) is allowed to be configured for each configured Xcp over Ethernet connection (when `XcpConnectionInterfaceType` = `XcpConnectionOverEthernet`). There is one deviation from this rule: If the Ethernet connection type is UDP/IP (`SoAdSocketProtocol`), and the wildcard is used for the configured socket connection (`SoAdSocketRemoteIpAddress` = `ANY` or `SoAdSocketRemotePort` = `0`), then two connections, one for each direction, must be defined for the same

Containers included		
		socket. Note: This workaround is required because the master can change the port address when it is still connected, and it is specified in ASAM that this case is allowed. ► If multiple Xcp PDUs are configured for one of the available directions (`XcpTxPduConnectionInfo`/`XcpRxPduConnectionInfo`), the SoAd PDU header (`SoAdPduHeaderEnable`) must be enabled. ► `SoAdSocketSoConModeChgNotification` must be: ► Enabled, if the socket type (`SoAdSocketProtocol`) is TCP (`SoAdSocketTcp`) ► Disabled, if the socket type (`SoAdSocketProtocol`) is UDP (`SoAdSocketUdp`) ► `SoAdSocketTcpInitiate` must be disabled for the used TCP/IP socket connection group (`SoAdSocketConnectionGroup`) - refer to the ASAM specification (the Xcp slave is listener).
XcpMemoryAccessArea	1..1	**Label:** Xcp Memory Access Areas

5.2.1.5. XcpDaqList

Containers included		
Container name	**Multiplicity**	**Description**
XcpDto	0..n	Defines Data Transfer Object (DTO) specific parameters for the DAQ list. Note: This container is not supported in the current implementation. The current implementation does not support the configuration of individual CAN IDs for each DAQ list. Thus, the configuration is not required as of now. All DTOs are transmitted with the same CAN ID.
XcpOdt	0..n	Defines Object Descriptor Table (ODT) specific parameters. Dependency on parameter(s):

Containers included		
		► Number of Predefined DAQ Lists (`XcpMinDaq`): ODTs can be configured only for a predefined DAQ list.

Parameters included	
Parameter name	**Multiplicity**
XcpDaqListNumber	1..1
XcpDaqListType	1..1
XcpMaxOdt	1..1
XcpMaxOdtEntries	1..1
XcpDaqEventFixed	1..1
XcpDaqListPriority	1..1
XcpDaqListPrescaler	1..1

Parameter Name	XcpDaqListNumber	
Label	DAQ List Number	
Description	Defines the index number of the DAQ list (ASAM parameter: `DAQ_LIST_NUMBER`). Within one and the same Xcp slave device, the range for the DAQ list number starts from 0 and must be continuous. A symbolic value (preprocessor macro) is also generated for each DAQ list number. Range: ► 0 .. 65534	
Multiplicity	1..1	
Type	INTEGER	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	XcpDaqListType
Label	DAQ List Type
Description	Defines the direction of the DAQ list (ASAM parameter: `DAQ_LIST_TYPE`). Range: ► `DAQ`: List shall be used only with direction DAQ.

	► STIM: List shall be used only with direction STIM. ► DAQ_STIM: List shall be used either with direction DAQ or with direction STIM. Dependency on parameter(s): ► STIM direction support (XcpStimSupported): must be enabled to configure this parameter as STIM or DAQ_STIM. ► Number of Predefined DAQ Lists (XcpMinDaq): this parameter is enabled only if the DAQ list is preconfigured.
Multiplicity	1..1
Type	ENUMERATION
Default value	DAQ
Range	DAQ DAQ_STIM STIM
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	AUTOSAR_ECUC

Parameter Name	**XcpMaxOdt**
Label	Total number of ODTs
Description	Defines the total number of Object Descriptor Tables (ODTs) in this DAQ list (ASAM parameter: MAX_ODT). Note that the WRITE_DAQ or WRITE_DAQ_MULITPLE commands must cover all the ODT's specified by this parameter. Range: ► 1 .. 252 Dependency on parameter(s): ► Number of Predefined DAQ Lists (XcpMinDaq), DAQ List Number (XcpDaqListNumber): XcpMaxOdt is ignored for predefined DAQ lists (XcpDaqListNumber less than XcpMinDaq) and instead the corresponding value is calculated. ► When the XcpSupportForOverloadMSB is enabled, not knowing the actual type of the DAQ list at run-time, the decreasing of the ODT range upper limit to '123' must be considered if the direction becomes 'DAQ' at run-time.

	Therefore, a warning is issued when this limit is exceeded. The following aspects are considered when the ODT range upper limit is calculated: ► If the identification field is ABSOLUTE, (see `XcpIdentification-FieldType`), all ODTs configured for the DAQ list with a lower ID than the selected one are considered. Also, all the ODTs with a lower ID than the selected one are considered, too, in the computation formula. ► If the identification field is RELATIVE[/_-], see `XcpIdentification-FieldType`, only the ODTs with a lower ID than the selected one are considered in the computation formula. ► When the `XcpSupportForOverloadMSB` is disabled, not knowing the actual type of the DAQ list at run-time, the decreasing of the ODT range upper limit to '191' must be considered if the direction becomes 'STIM' at run-time. Therefore, a warning is issued when this limit is exceeded. The following aspects are considered when the ODT range upper limit is calculated: ► If the identification field is 'ABSOLUTE', see `XcpIdentification-FieldType`, all ODTs configured for the DAQ list with a lower ID than the selected one are considered. Also, all the ODTs with a lower ID than the selected one are considered, too, in the computation formula. ► If the identification field is 'RELATIVE[/_-]', see `XcpIdentification-FieldType`, only the ODTs with a lower ID than the selected one are considered in the computation formula. **Optimization Effect:** ► **ROM reduction (config):** Selecting a smaller value for this parameter reduces the ROM consumption of the module configuration. ► **RAM reduction (config):** Selecting a smaller value for this parameter reduces the RAM consumption of the module configuration. ► **Execution time reduction (code):** Selecting a smaller value for this parameter reduces the execution time of the module code.
Multiplicity	1..1
Type	INTEGER
Default value	1
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	AUTOSAR_ECUC

Parameter Name	**XcpMaxOdtEntries**
Label	Maximum ODT Entries Per ODT

Description	Defines the maximum number of entries into an Object Descriptor Table (ODT) of the DAQ list if the current configuration is DAQ_STATIC (ASAM parameter: `MAX_ODT_ENTRIES`). Range: ► 1 .. 255 Dependency on parameter(s): ► Number of Predefined DAQ Lists (`XcpMinDaq`), DAQ List Number (`XcpDaqListNumber`): `XcpMaxOdtEntries` is ignored for predefined DAQ lists (XcpDaqListNumber less than XcpMinDaq) and instead the corresponding value is calculated. **Optimization Effect:** ► **ROM reduction (config):** Selecting a smaller value for this parameter reduces the ROM consumption of the module configuration. ► **RAM reduction (config):** Selecting a smaller value for this parameter reduces the RAM consumption of the module configuration. ► **Execution time reduction (code):** Selecting a smaller value for this parameter reduces the execution time of the module code.
Multiplicity	1..1
Type	INTEGER
Default value	1
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	AUTOSAR_ECUC

Parameter Name	**XcpDaqEventFixed**
Label	DAQ Event Fixed
Description	Defines whether the event channel for this DAQ list can be changed or not. Dependency on parameter(s): ► Number of predefined DAQ lists (`XcpMinDaq`): This parameter is enabled only if the DAQ list is preconfigured.
Multiplicity	1..1
Type	BOOLEAN
Default value	false

Configuration class	**PreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**XcpDaqListPriority**
Label	DAQ List Priority
Description	Defines the priority of the DAQ list when the slave processes the different DAQ lists. The DAQ list with *XcpDaqListPriority* = 255 has the highest priority. Range: ► 0 .. 255 Dependency on parameter(s): ► Number of predefined DAQ lists (`XcpMinDaq`): This parameter is enabled only if the DAQ list is preconfigured.
Multiplicity	1..1
Type	INTEGER
Default value	0
Range	<=255 >=0
Configuration class	**PreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpDaqListPrescaler**
Label	DAQ List Prescaler
Description	Defines the prescaler value of the DAQ list. For a transmission rate without reduction, the prescaler must be 1. For a reduced transmission rate, the prescaler must be greater than 1. The prescaler is only used for DAQ lists with DIRECTION = DAQ. Range: ► 1 .. 255 Dependency on parameter(s):

	▶ Number of predefined DAQ lists (`XcpMinDaq`): This parameter is enabled only if the DAQ list is preconfigured.
Multiplicity	1..1
Type	INTEGER
Default value	1
Range	<=255 >=1
Configuration class	**PreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

5.2.1.6. XcpDto

Parameters included	
Parameter name	**Multiplicity**
XcpDtoPid	1..1
XcpDto2PduMapping	1..1

Parameter Name	**XcpDtoPid**
Description	This parameter indicates the Packet Identifier (PID) that identifies the content of the DTO, namely the ODT, within a DAQ list. Note: This parameter is not supported in the current implementation. This parameter is disabled, because the value for PID shall be generated from the ODT index.
Multiplicity	1..1
Type	INTEGER
Default value	0
Range	<=251 >=0
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	AUTOSAR_ECUC

Parameter Name	**XcpDto2PduMapping**
Label	DTO to PDU mapping

Description	This reference specifies the mapping of the DTO to the PDUs from the lower-layer interfaces (CanIf, FrIf, SoAd, and CDD). Note: This parameter is not supported in the current implementation.	
Multiplicity	1..1	
Type	CHOICE-REFERENCE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

5.2.1.7. XcpOdt

Containers included		
Container name	**Multiplicity**	**Description**
XcpOdtEntry	1..n	Defines the configuration parameters to configure an ODT entry.

Parameters included	
Parameter name	**Multiplicity**
XcpOdtEntryMaxSize	1..1
XcpOdtNumber	1..1
XcpOdt2DtoMapping	1..1

Parameter Name	**XcpOdtEntryMaxSize**
Label	ODT Entry Maximum Size
Description	Defines the upper limit for the size of the element described by an Object Descriptor Table (ODT) entry. Note: This parameter is not supported in the current implementation. The current implementation uses XcpGeneral/XcpOdtEntrySizeDaq and XcpGeneral/XcpOdtEntrySizeStim instead of this parameter. Depending on the DAQ list type that this Object Descriptor Table (ODT) belongs to, it describes the limit for a DAQ (MAX_ODT_ENTRY_SIZE_DAQ) or a STIM (MAX_ODT_ENTRY_SIZE_STIM). Range: ► 0 .. 254

Multiplicity	1..1	
Type	INTEGER	
Default value	0	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**XcpOdtNumber**
Label	ODT Number
Description	Defines the index number of this Object Descriptor Table (ODT) within the DAQ list (ASAM parameter: `ODT_NUMBER`). The scope of this parameter is local for a DAQ list and ODT numbers must be unique within one and the same slave device. For every DAQ list, the numbering of the Object Descriptor Tables (ODTs) through *ODT_NUMBER* restarts from 0. Range: ► When the `XcpSupportForOverloadMSB` is enabled, if the DAQ list direction is `DAQ` or `DAQ_STIM`, the ODT range upper limit decreases to 123 (see ASAM specifications). The following aspects are considered when the ODT range upper limit is calculated: ► 0 .. (123 - number of configured ODTs): when `XcpIdentificationFieldType` = `ABSOLUTE`: all ODTs configured for the DAQ list with a lower ID than the selected one are considered. Also, all the ODTs with a lower ID than the selected one are considered, too, in the computation formula. ► 0 .. 123: when `XcpIdentificationFieldType` != `ABSOLUTE`: only the ODTs with a lower ID than the selected one are considered in the computation formula. ► When the `XcpSupportForOverloadMSB` is disabled, if the DAQ list direction is `STIM` or `DAQ_STIM`, the ODT range upper limit decreases to 191 (see ASAM specifications). The following aspects are considered when the ODT range upper limit is calculated: ► 0 .. (191 - number of configured ODTs): when `XcpIdentificationFieldType` = `ABSOLUTE`: all ODTs configured for the DAQ list with a lower ID than the selected one are considered. Also, all the ODTs with a lower ID than the selected one are considered, too, in the computation formula.

	▶ 0 .. 191: when `XcpIdentificationFieldType` != `ABSOLUTE`: only the ODTs with a lower ID than the selected one are considered in the computation formula.
Multiplicity	1..1
Type	INTEGER
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	AUTOSAR_ECUC

Parameter Name	**XcpOdt2DtoMapping**
Label	ODT to DTO mapping
Description	This reference maps the Object Descriptor Table (ODT) to the corresponding DTO in which it is transmitted. Note: This parameter is not supported in the current implementation. The current implementation does not support the configuration of individual CAN IDs for each DAQ list. Thus, the configuration is not required as of now. All DTOs are transmitted with the same CAN ID.
Multiplicity	1..1
Type	REFERENCE
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	AUTOSAR_ECUC

5.2.1.8. XcpOdtEntry

Parameters included	
Parameter name	**Multiplicity**
XcpOdtEntryAddress	1..1
XcpOdtEntryLength	1..1
XcpOdtEntryNumber	1..1
XcpOdtEntryBitOffset	0..1

Parameter Name	**XcpOdtEntryAddress**
Label	ODT Entry Address
Description	Defines the memory address to a measurement object that is part of the Object Descriptor Table (ODT).

	Enter here a valid C-expression that evaluates to an address in the slave's memory, e.g. `&VariableName` for a variable address or `0xEB15C001` for a literal address. Any literal address must point to a valid address into the slave's memory. If variable names are used, the Xcp module expects all required symbol declarations and definitions to be available after including the User Header File, which is specified by the configuration parameter XcpGeneral/XcpUserHeader.
Multiplicity	1..1
Type	LINKER-SYMBOL
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	AUTOSAR_ECUC

Parameter Name	**XcpOdtEntryLength**
Label	ODT Entry Length
Description	Defines the length of the memory area that is referenced by the Object Descriptor Table (ODT) entry. Range: ► 1 .. 255 Note: This parameter is counted in address granularity (AG, parameter XcpAddressGranularity) bytes.
Multiplicity	1..1
Type	INTEGER
Default value	1
Range	<=255 >=1
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	AUTOSAR_ECUC

Parameter Name	**XcpOdtEntryNumber**
Label	ODT Entry Number
Description	Defines the index number of the Object Descriptor Table (ODT) entry (ASAM parameter: `ODT_ENTRY_NUMBER`). Range: ► 0 .. 254

Multiplicity	1..1	
Type	INTEGER	
Range	<=254 >=0	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**XcpOdtEntryBitOffset**	
Label	ODT Entry Bit Offset	
Description	Defines the bit offset if the element represents a status bit. Range: ► 0 .. 31 Dependency on parameter(s): ► Address Granularity (`XcpAddressGranularity`): If Address Granularity is BYTE, the maximum value of this parameter is 7. If Address Granularity is WORD, the maximum value of this parameter is 15. If Address Granularity is DWORD, the maximum value of this parameter is 31. ► ODT Entry Length (`XcpOdtEntryLength`): must be 1 if Bit Offset is enabled. ► XcpEnableBitOffset: If XcpEnableDaqOptimization is TRUE and XcpEnableBitOffset is FALSE then XcpOdtEntryBitOffset cannot be used.	
Multiplicity	0..1	
Type	INTEGER	
Default value	0	
Configuration class	**PreCompile:**	VariantPreCompile

5.2.1.9. XcpDemEventParameterRefs

Parameters included	
Parameter name	**Multiplicity**
XCP_E_INIT_FAILED	0..1
XCP_E_RETRY_FAILED	1..1

Parameters included	
XCP_E_PDU_BUFFER_LOCKED	1..1
XCP_E_RESP_CTO_QUEUE_FULL	1..1
XCP_E_PDU_LOST	1..1
XCP_E_STIMULATION_DATA_LOST	1..1
XCP_E_ACQUISITION_DATA_LOST	1..1
XCP_E_PDU_BUFFER_FULL	1..1
XCP_E_ILLEGAL_MEMORY_ACCESS	1..1

Parameter Name	XCP_E_INIT_FAILED	
Label	Error for failed initialization	
Description	Reference to the `DemEventParameter` that shall be issued when the error *Initialization of XCP failed* has occurred. Note: This parameter is not supported in the current implementation.	
Multiplicity	0..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	XCP_E_RETRY_FAILED
Label	Error for Failed Transmission Retry
Description	Reference to the DemEventParameter that shall be issued when the error *Tx retry failed* occurs. Dependency on parameter(s): ► Xcp Retry Failed Reporting Type (`XcpRetryFailedReportToDem`): Select DEM to enable the reporting of `XCP_E_RETRY_FAILED`. Further notes: ► Activation: Thrown if the XCP message transmission (including the configured number of retries) fails. ► Healing: None. The error resides in memory until it is deleted. ► Trigger debounce: XCP_E_RETRY_FAILED shall be thrown only after a maximum number of retries (`XcpTxRetryCount`) is exceeded for requests configured with a retry counter.

	► Rate of diagnostic checks: Checked on every call of the service that reports this error. A list of API functions that report this error can be found in the table of production errors in the Integration notes section of the module references.
Multiplicity	1..1
Type	REFERENCE
Configuration class	**PreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XCP_E_PDU_BUFFER_LOCKED**
Label	Error for Pdu Buffer Locked
Description	Reference to the DemEventParameter that shall be issued when the error *PDU buffer locked* occurs. Dependency on parameter(s): ► Xcp Pdu Buffer Locked Reporting Type (`XcpPduBufferLockedReportToDem`): Select DEM to enable the reporting of `XCP_E_PDU_BUFFER_LOCKED`. Further notes: ► Activation: Thrown if the internal PDU ID is not mapped correctly to an Xcp connection ID. ► Healing: None. The error resides in memory until it is deleted. ► Trigger debounce: None. The error is reported on first occurrence. ► Rate of diagnostic checks: Checked on every call of the service that reports this error. A list of API functions that report this error can be found in the table of production errors in the Integration notes section of the module references.
Multiplicity	1..1
Type	REFERENCE
Configuration class	**PreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XCP_E_RESP_CTO_QUEUE_FULL**
Label	Error for Response CTO Queue Full
Description	Reference to the DemEventParameter that shall be issued when the error *Response CTO queue full* occurs.

	Dependency on parameter(s): ► Xcp Response CTO Queue Full Reporting Type (`XcpRespCTOQueueFull-ReportToDem`): Select DEM to enable the reporting of `XCP_E_RESP_-CTO_QUEUE_FULL`. Further notes: ► Activation: Thrown if the response CTO queue, which is used to enqueue the response packet, is full. ► Healing: None. The error resides in memory until it is deleted. ► Trigger debounce: None. The error is reported on first occurrence. ► Rate of diagnostic checks: Checked on every call of the service that reports this error. A list of API functions that report this error can be found in the table of production errors in the Integration notes section of the module references.
Multiplicity	1..1
Type	REFERENCE
Configuration class	**PreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XCP_E_PDU_LOST**
Label	Error for PDU Lost
Description	Reference to the DemEventParameter that shall be issued when the error *PDU lost* occurs. Dependency on parameter(s): ► Xcp PDU Lost Reporting Type (`XcpPDULostReportToDem`): Select DEM to enable the reporting of `XCP_E_PDU_LOST`. Further notes: ► Activation: Thrown if an XCP message is received, but it is discarded by XCP because it is detected as incorrect. ► Healing: None. The error resides in memory until it is deleted. ► Trigger debounce: None. The error is reported on first occurrence. ► Rate of diagnostic checks: Checked on every call of the service that reports this error. A list of API functions that report this error can be found in the table of production errors in the Integration notes section of the module references.

Multiplicity	1..1	
Type	REFERENCE	
Configuration class	**PreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**XCP_E_STIMULATION_DATA_LOST**
Label	Error for Stimulation Data Lost
Description	Reference to the DemEventParameter that shall be issued when the error *Stimulation data lost* occurs. Dependency on parameter(s): ► Xcp Stimulation Data Lost Reporting Type (`XcpStimulationDataLostReportToDem`): Select DEM to enable the reporting of `XCP_E_STIMULATION_DATA_LOST`. Further notes: ► Activation: The error is triggered by Xcp_MainFunction()/Xcp_MainFunction_[EventName]/ Xcp_<If>RxIndication() if, when unpacking PDUs, at least one PDU message is silently lost. That means there is no notification towards the Xcp master. ► Healing: None. The error resides in memory until it is deleted. ► Trigger debounce: None. The error is reported on first occurrence. ► Rate of diagnostic checks: Checked on every call of the service that reports this error. A list of API functions that report this error can be found in the table of production errors in the Integration notes section of the module references.
Multiplicity	1..1
Type	REFERENCE
Configuration class	**PreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XCP_E_ACQUISITION_DATA_LOST**
Label	Error for Acquisition Data Lost
Description	Reference to the DemEventParameter that shall be issued when the error *Acquisition data lost* occurs. Dependency on parameter(s):

	► Xcp Acquisition Data Lost Reporting Type (`XcpAcquisitionDataLostReportToDem`): Select DEM to enable the reporting of `XCP_E_ACQUISITION_DATA_LOST`. Further notes: ► Activation: The error is triggered by Xcp_MainFunction()/Xcp_MainFunction_[EventName]/ Xcp_<If>RxIndication() if, when unpacking PDUs, at least one PDU message is silently lost. That means there is no notification towards the Xcp master. ► Healing: None. The error resides in memory until it is deleted. ► Trigger debounce: None. The error is reported on first occurrence. ► Rate of diagnostic checks: Checked on every call of the service that reports this error. A list of API functions that report this error can be found in the table of production errors in the Integration notes section of the module references.
Multiplicity	1..1
Type	REFERENCE
Configuration class	**PreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XCP_E_PDU_BUFFER_FULL**
Label	Error for Response PDU Buffer Full
Description	Reference to the DemEventParameter that shall be issued when the error *PDU buffer full* occurs. Dependency on parameter(s): ► PDU Buffer Full Reporting Type (`XcpPDUBufferFullReportToDem`): Select DEM to enable the reporting of `XCP_E_PDU_BUFFER_FULL`. Further notes: ► Activation: The error is triggered by Xcp_<If>RxIndication() if the L-PDU buffer corresponding to the received PDU is full. This may happen if the Xcp main function is called with a higher latency than the lowest latency used by the master to send messages. ► Healing: None. The error resides in memory until it is deleted. ► Trigger debounce: None. The error is reported on first occurrence.

	▶ Rate of diagnostic checks: Checked on every call of the service that reports this error. A list of API functions that report this error can be found in the table of production errors in the Integration notes section of the module references.
Multiplicity	1..1
Type	REFERENCE
Configuration class	**PreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XCP_E_ILLEGAL_MEMORY_ACCESS**
Label	Error for Illegal Memory Access
Description	Reference to the DemEventParameter that shall be issued when the error *Illegal memory access* occurs. Dependency on parameter(s): ▶ Xcp Illegal Memory Access Reporting Type (`XcpIllegalMemoryAccessReportToDem`): Select DEM to enable the reporting of `XCP_E_ILLEGAL_MEMORY_ACCESS`. Further notes: ▶ Activation: Thrown if an illegal memory access is performed. ▶ Healing: None. The error resides in memory until it is deleted. ▶ Trigger debounce: None. The error is reported on first occurrence. ▶ Rate of diagnostic checks: Checked on every call of the service that reports this error. A list of API functions that report this error can be found in the table of production errors in the Integration notes section of the module references.
Multiplicity	1..1
Type	REFERENCE
Configuration class	**PreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

5.2.1.10. XcpEventChannel

Parameters included	
Parameter name	**Multiplicity**

Parameters included	
XcpEventChannelConsistency	1..1
XcpEventChannelMaxDaqList	1..1
XcpEventChannelNumber	1..1
XcpEventChannelPriority	1..1
XcpEventChannelTimeCycle	1..1
XcpEventChannelTimeUnit	0..1
XcpEventChannelType	1..1
XcpEventChannelTriggeredDaqListRef	0..n

Parameter Name	XcpEventChannelConsistency
Label	Event Channel Consistency
Description	Defines the consistency of the event channel. Range: ► `ODT`: The function Xcp_MainFunction() shall process one ODT associated with the event in one cyclic call. ► `DAQ`: The function Xcp_MainFunction() shall process one DAQ list associated with the event in one cyclic call. ► `EVENT`: The function Xcp_MainFunction() shall process all DAQ lists associated with the event in one cyclic call. **Optimization Effect:** ► **RAM reduction (config):** Setting this parameter, together with all other event channel consistency, to a value of ODT reduces the RAM consumption of the module configuration. Note: Xcp internal buffers grow a lot if the overall event consistency is EVENT or DAQ. If RAM consumption issues are found, consider changing overall event consistency to ODT. RAM consumption issues are most likely to occur when Xcp on Ethernet (`XcpOnEthernetEnabled`) is enabled because the MAX_DTO can be much bigger than the CAN or FlexRay case. The overall event consistency is calculated as: ► If a single event consistency is set to EVENT, the overall event consistency is set to EVENT. ► If a single event consistency is set to DAQ and no event has event consistency set to EVENT, the overall event consistency is set to DAQ.

	► If all events have event consistency ODT, the overall event consistency is set to ODT.
Multiplicity	1..1
Type	ENUMERATION
Default value	ODT
Range	DAQ EVENT ODT
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	AUTOSAR_ECUC

Parameter Name	**XcpEventChannelMaxDaqList**
Label	Maximum Number of DAQ Lists
Description	Defines the maximum number of DAQ lists that are handled by this event channel (ASAM parameter: `MAX_DAQ_LIST`). Range: ► 1 .. 255 **Optimization Effect:** ► **RAM reduction (config):** Selecting a smaller value for this parameter reduces the RAM consumption of the module configuration.
Multiplicity	1..1
Type	INTEGER
Default value	1
Range	<=255 >=1
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	AUTOSAR_ECUC

Parameter Name	**XcpEventChannelNumber**
Label	Event Channel Number
Description	Defines the index number of the event channel (ASAM parameter: `EVENT_CHANNEL_NUMBER`).

	The event channel number specifies the generic signal source that effectively determines the data transmission timing. A symbolic value (preprocessor macro) is also generated for each event channel number. Range: ► 0 .. 65534
Multiplicity	1..1
Type	INTEGER
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	AUTOSAR_ECUC

Parameter Name	**XcpEventChannelPriority**
Label	Event Channel Priority
Description	Defines the priority of the event channel when the slave processes the different event channels (ASAM parameter: `EVENT_CHANNEL_PRIORITY`). This prioritization is a fixed attribute of the slave and therefore read-only. The event channel with priority 255 has the highest priority. Range: ► 0 .. 255
Multiplicity	1..1
Type	INTEGER
Default value	0
Range	<=255 >=0
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	AUTOSAR_ECUC

Parameter Name	**XcpEventChannelTimeCycle**
Label	Event Channel Time Cycle
Description	Defines the sampling period for this event in multiples of the event channel time unit (ASAM parameter: `EVENT_CHANNEL_TIME_CYCLE`). Range:

	► 0 .. 255 Note: ► Events with a sampling period greater than 0 are automatically sampled by the function `Xcp_MainFunction()`. ► Events with a sampling period of 0 are not automatically sampled. Use the API function `Xcp_SetEvent()` to trigger those events. The events are then handled in the next execution of `Xcp_MainFunction()`. ► `Xcp_SetEvent()` may not be called for events with a sampling period different than 0. Dependency on parameter(s): ► Event Channel Time Unit (`XcpEventChannelTimeUnit`): the sampling period is defined in multiples of the event channel time unit. ► Main Function Period [s] (`XcpMainFunctionPeriod`): the sampling period (i.e. Event Channel Time Cycle * Event Channel Time Unit) shall be an integer multiple of the main function period. **Optimization Effect:** ► **ROM reduction (code):** Setting the value ZERO for this parameter reduces the ROM consumption of the module code. ► **Execution time reduction (code):** Setting the value ZERO for this parameter reduces the execution time of the module code.
Multiplicity	1..1
Type	INTEGER
Default value	0
Range	<=255 >=0
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	AUTOSAR_ECUC

Parameter Name	**XcpEventChannelTimeUnit**
Label	Event Channel Time Unit
Description	Defines the time unit for calculating the sampling period of this event (ASAM parameter: `EVENT_CHANNEL_TIME_UNIT`). Range:

	▶ TIMESTAMP_UNIT_1NS: 1ns (nanosecond) ▶ TIMESTAMP_UNIT_10NS: 10ns ▶ TIMESTAMP_UNIT_100NS: 100ns ▶ TIMESTAMP_UNIT_1US: 1us (microsecond) ▶ TIMESTAMP_UNIT_10US: 10us ▶ TIMESTAMP_UNIT_100US: 100us ▶ TIMESTAMP_UNIT_1MS: 1ms (millisecond) ▶ TIMESTAMP_UNIT_10MS: 10ms ▶ TIMESTAMP_UNIT_100MS: 100ms ▶ TIMESTAMP_UNIT_1S: 1s (second) ▶ TIMESTAMP_UNIT_1PS: 1ps (picosecond) ▶ TIMESTAMP_UNIT_10PS: 10ps ▶ TIMESTAMP_UNIT_100PS: 100ps Dependency on parameter(s): ▶ Event Channel Time Cycle (XcpEventChannelTimeCycle): must be set to a value different from 0 to allow setting the event channel time unit. ▶ Main Function Period [s] (XcpMainFunctionPeriod): the sampling period (i.e. Event Channel Time Cycle * Event Channel Time Unit) shall be an integer multiple of the main function period.
Multiplicity	0..1
Type	ENUMERATION
Default value	TIMESTAMP_UNIT_1MS
Range	TIMESTAMP_UNIT_100MS TIMESTAMP_UNIT_100NS TIMESTAMP_UNIT_100PS TIMESTAMP_UNIT_100US TIMESTAMP_UNIT_10MS TIMESTAMP_UNIT_10NS TIMESTAMP_UNIT_10PS TIMESTAMP_UNIT_10US TIMESTAMP_UNIT_1MS TIMESTAMP_UNIT_1NS

	TIMESTAMP_UNIT_1PS	
	TIMESTAMP_UNIT_1S	
	TIMESTAMP_UNIT_1US	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**XcpEventChannelType**
Label	Event Channel Type
Description	Defines the kind of DAQ list that can be allocated to this event channel (ASAM parameter: `EVENT_CHANNEL_TYPE`). Range: ► `DAQ`: List allocated shall be only with direction DAQ. ► `STIM`: List allocated shall be only with direction STIM. ► `DAQ_STIM`: List allocated shall be either with direction DAQ or with direction STIM.
Multiplicity	1..1
Type	ENUMERATION
Default value	DAQ
Range	DAQ DAQ_STIM STIM
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	AUTOSAR_ECUC

Parameter Name	**XcpEventChannelTriggeredDaqListRef**
Label	DAQ List Reference
Description	Reference to the predefined DAQ lists that are triggered by this event channel.
Multiplicity	0..n
Type	REFERENCE
Range	node:when(node:value(../../XcpEventChannelType) = 'DAQ_STIM', node:paths(../../../../XcpDaqList/*[XcpDaqListType]), '') node:when(node:value(../../XcpEventChannelType) = 'STIM', node:paths(../../../../XcpDaqList/*[XcpDaqListType = 'STIM' or XcpDaqListType = 'DAQ_STIM']), '')

	node:when(node:value(../../XcpEventChannelType) = 'DAQ', node:paths(../../../../XcpDaqList/*[XcpDaqListType = 'DAQ' or XcpDaqListType = 'DAQ_STIM']), '')	
Configuration class	**PreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

5.2.1.11. XcpPdu

Containers included		
Container name	**Multiplicity**	**Description**
XcpRxPdu	1..1	Defines the Rx PDUs. Dependency on parameter(s): ► It is not allowed to configure an `XcpRxPdu` that is not further assigned to a connection, refer to `XcpMapRxPdu2Connection`.
XcpTxPdu	1..1	**Label:** Tx PDU Id Defines the Tx PDUs. Dependency on parameter(s): ► It is not allowed to configure an `XcpTxPdu` that is not further assigned to a connection, refer to `XcpMapTxPdu2Connection`.

5.2.1.12. XcpRxPdu

Parameters included	
Parameter name	**Multiplicity**
XcpRxPduId	1..1
XcpRxPduRef	1..1

Parameter Name	**XcpRxPduId**
Label	Rx PDU Id
Description	Defines the ID of the PDU that is received via an Xcp_<module>RxIndication. A symbolic value (preprocessor macro) is also generated for each PDU ID.

	The Xcp's Rx PDU IDs must be zero-based and consecutive. Range: 0 .. 65535
Multiplicity	1..1
Type	INTEGER
Range	<=65535 >=0
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	AUTOSAR_ECUC

Parameter Name	**XcpRxPduRef**
Label	Rx PDU Reference
Description	Reference to the external Rx PDU definition in the EcuC module. Dependency on parameter(s): ► A valid reference to EcuC's `Pdu` parameters must be provided. ► A unique EcuC PDU must be provided as reference for each Xcp Rx PDU. ► It is not allowed to configure bidirectional Xcp PDUs (the same PDU cannot be given as reference for both `XcpRxPdu` and `XcpTxPdu`).
Multiplicity	1..1
Type	REFERENCE
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	AUTOSAR_ECUC

5.2.1.13. XcpTxPdu

Parameters included	
Parameter name	**Multiplicity**
XcpTxPduId	1..1
XcpTxPduSupportForCddTriggerTransmit	1..1
XcpTxPduRef	1..1

Parameter Name	**XcpTxPduId**
Description	Defines the PDU identifier, which must be used by the lower layer BSW module for TxConfirmations or TriggerTransmits.

	A symbolic value (preprocessor macro) is also generated for each PDU ID. The Xcp's Tx PDU IDs must be zero-based and consecutive. Range: 0 .. 65535
Multiplicity	1..1
Type	INTEGER
Range	<=65535 >=0
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	AUTOSAR_ECUC

Parameter Name	**XcpTxPduSupportForCddTriggerTransmit**
Label	PDU Support for CDD Trigger Transmit
Description	Enables the Tx PDU to support the TriggerTransmit functionality over a CDD connection. Dependency on parameter(s): The XcpOnCddEnabled parameter shall be set in order for this parameter to be used.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**PreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpTxPduRef**
Label	Tx PDU Reference
Description	Reference to the external Tx PDU definition in the EcuC module. Dependency on parameter(s): ► A valid reference to EcuC's `Pdu` parameters must be provided. ► A unique EcuC PDU must be provided as reference for each Xcp Tx PDU. ► It is not allowed to configure bidirectional Xcp PDUs (the same PDU cannot be given as reference for both `XcpTxPdu` and `XcpRxPdu`).
Multiplicity	1..1

Type	REFERENCE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

5.2.1.14. XcpConnectionCfg

Containers included		
Container name	**Multiplicity**	**Description**
XcpConnectionInterfaceType	1..1	**Label:** Xcp Connection Interface Type Choice container used to select the connection type: ► `XcpConnectionOverCAN`: only PDUs mapped to the referenced CanIf configuration container are accepted to the selected connection configuration ► `XcpConnectionOverCANFD`: only PDUs mapped to the referenced CanIf configuration container are accepted to the selected connection configuration ► `XcpConnectionOverFlexRay`: only PDUs mapped to the same FlexRay controller are accepted to the selected connection configuration. ► `XcpConnectionOverEthernet`: only PDUs mapped to the same Socket connection group are accepted to the selected connection configuration. ► `XcpConnectionOverCDD`: only PDUs referenced from a CDD configuration are accepted to the selected connection configuration.
XcpA2LGenerationSupport	1..1	**Label:** A2L Generation support (CAN/CAN-FD configuration) Contains various configuration parameters that are used to generate the A2L file.
XcpTxPduConnectionInfo	1..n	Groups PDU information, for transmit PDU, related to the selected connection. Dependency on parameter(s): ► Unusable PDUs are not allowed to be configured. At least one of the following parameters must be enabled in order to make the PDU usable:

Containers included		
		▶ XcpPduSupportForResErrCtoType: enables transmitting RES/ERR packages via the selected PDU ▶ XcpPduSupportForEvServCtoType: enables transmitting EV/SERV packages via the selected PDU ▶ XcpPduSupportForDaqDtoType: enables transmitting DAQ DTO packages via the selected PDU ▶ It is required, for each connection, to have just one PDU supporting RES/ERR packages (refer to XcpPduSupportForResErrCtoType parameter). ▶ If the sending event packages feature is enabled (XcpEventPacketEnabled), it is required, for each connection, to have at least one PDU supporting EV/SERV packages (refer to XcpPduSupportForEvServCtoType parameter). ▶ If the processing DAQ list feature is enabled (XcpDaqSupported), it is required, for each connection, to have at least one PDU supporting DAQ DTO packages (refer to XcpPduSupportForDaqDtoType parameter). ▶ Dependencies on XcpConnectionInterfaceType parameter: ▶ It is not allowed to share Xcp Tx PDUs between connections. ▶ Each Xcp PDU mapped to a connection must be referenced only once into the corresponding transport layer interface, towards Xcp. In case of Xcp connection over Ethernet, the PDU cannot be shared with another application. ▶ For each connection, it is allowed to configure just one PDU supporting RES/ERR CTO data (refer to XcpPduSupportForEvServCtoType). ▶ For each connection, it is allowed to configure at most one PDU supporting EV/SERV CTO data (refer to XcpPduSupportForEvServCtoType). ▶ For each connection, it is allowed to configure any number of PDUs supporting DAQ DTO packages

Containers included		
		(refer to `XcpPduSupportForDaqDtoType`). But there is one exception to this rule: If the parameter `XcpSupportForGetDaqId` is enabled, and the selected connection is over CAN, then at most one PDU supporting DAQ DTO data is allowed to be configured. ► If the Xcp connection is over FlexRay, the following transport layer specific dependency must be accomplished: ► `FrIfNoneMode`: must be disabled (the transmission is always triggered by Xcp, by calling <If>_Transmit()) ► If the Xcp connection is over Ethernet, the following transport layer specific dependencies must be accomplished: ► `SoAdSocketConnection`: It is not allowed to share the same Tx SoAd connection between two Xcp connections over the Ethernet. ► `SoAdTxUpperLayerType`: must be equal to `IF` (transport protocol is not supported). ► `SoAdPduRouteDest`: It is allowed to have only one child. (To prevent the connection of two clients at the same time, the PDU cannot be shared to multiple socket connections.) ► `SoAdTxUdpTriggerMode`: must be disabled (only immediate transmit is supported for Ethernet transport layer). ► All PDUs belonging to the same connection must belong to the same socket connection.
XcpRxPduConnectionInfo	1..n	Groups PDU information, for receive PDU, related to the selected connection. Dependency on parameter(s): ► Unusable PDUs are not allowed to be configured. At least one of the following parameters must be enabled in order to make the PDU usable:

Containers included		
		▶ XcpPduSupportForCmdCtoType: enables receiving CMD packages via the selected PDU ▶ XcpPduSupportForStimDtoType: enables receiving STIM DTO packages via the selected PDU ▶ A unique XcpRxLowerLayerHandleId must be configured per connection. ▶ It is required, for each connection, to have just one PDU supporting CMD packages (refer to XcpPduSupportForCmdCtoType parameter). ▶ If processing DAQ list with direction STIM is enabled (XcpStimSupported), it is required, for each connection, to have at least one PDU supporting STIM DTO packages (refer to XcpPduSupportForStimDtoType parameter). ▶ If a TCP/IP connection over Ethernet is configured, only one Rx PDU is allowed to be configured for each connection of this type. The Tx and Rx PDUs configured for a TCP/IP connection must belong to the same socket. ▶ Dependencies on XcpConnectionInterfaceType parameter: ▶ It is not allowed to share Xcp PDUs between connections. ▶ Each Xcp PDU mapped to a connection must be referenced, only once, in the corresponding transport layer interface, towards Xcp. With Xcp connection over Ethernet, the PDU cannot be shared with another application. ▶ For each connection, it is allowed to have at most one PDU supporting CMD CTO packages (refer to XcpPduSupportForEvServCtoType), used for receiving commands from the master. There is one exception: For Xcp over CAN, it is allowed to have a maximum of two PDUs supporting CMD CTOs: ▶ one PDU used to receive all commands but the broadcasted GET_SLAVE_ID (refer to XcpCtoSlavePduRef)

Containers included		
		▶ one PDU used to receive only the broadcasted GET_SLAVE_ID command (refer to `XcpBroadcastPduRef`). ▶ For each connection, it is allowed to configure any number of PDUs supporting STIM DTO packages. There is one exception: If the parameter `XcpSupportForGetDaqId` is enabled, and the selected connection is over CAN, then at most one PDU, supporting STIM DTO data is allowed to be configured (refer to `XcpPduSupportForEvServCtoType`). ▶ If the Xcp connection is over CAN, the following transport layer specific dependency must be accomplished: ▶ `CanIfRxPduCanId`: must be configured for the referenced Xcp Rx PDU. ▶ For each connection, it is allowed to have at most one PDU supporting CMD CTO packages (refer to `XcpPduSupportForEvServCtoType`), used for receiving commands from the master. There is one exception: For Xcp over CAN-FD, it is allowed to have a maximum of two PDUs supporting CMD CTOs: ▶ one PDU used to receive all commands but the broadcasted GET_SLAVE_ID (refer to `XcpCanFdCtoSlavePduRef`) ▶ one PDU used to receive only the broadcasted GET_SLAVE_ID command (refer to `XcpCanFdBroadcastPduRef`). ▶ For each connection, it is allowed to configure any number of PDUs supporting STIM DTO packages. There is one exception: If the parameter `XcpSupportForGetDaqId` is enabled, and the selected connection is over CAN-FD, then at most one PDU, supporting STIM DTO data is allowed to be configured (refer to `XcpPduSupportForEvServCtoType`).

Containers included		
		▶ If the Xcp connection is over CAN-FD, the following transport layer specific dependency must be accomplished: ▶ `CanIfRxPduCanId`: must be configured for the referenced Xcp Rx PDU. ▶ If the Xcp connection is over Ethernet, the following transport layer specific dependencies must be accomplished: ▶ `SoAdSocketConnection`: It is not allowed to share the same Tx SoAd connection between two Xcp connections over the Ethernet. ▶ `SoAdRxUpperLayerType`: must be equal to `IF` (transport protocol is not supported). ▶ `SoAdSocketRouteDest`: it is allowed to have only one child (to prevent broadcasting data over several PDUs). ▶ All PDUs belonging to the same connection must belong to the same socket connection.

Parameters included	
Parameter name	**Multiplicity**
XcpConnectionId	1..1
XcpConnectionApplicationRef	1..1
XcpMaxCto	1..1
XcpMaxDto	1..1
XcpMaxBS	1..1
XcpMaxCtoPgm	1..1
XcpMaxBSPgm	1..1
XcpTimestampEnabled	1..1

Parameter Name	XcpConnectionId
Label	Xcp Connection Id
Description	Xcp connection unique identifier. It is used to determine the connection that is used by the Xcp slave to communicate with the master.

	Range: ► 0 .. 254 ► 255: used to define an invalid connection ID	
Multiplicity	1..1	
Type	INTEGER	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**XcpConnectionApplicationRef**	
Label	Xcp Connection Application	
Description	Xcp reference to an Os Application. If the Bsw Distribution feature is enabled, each Xcp connection needs to be mapped to an Os Application. Xcp Bsw Distribution generates one Xcp instance per core. Each instance is represented by one or more Xcp connections.	
Multiplicity	1..1	
Type	REFERENCE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**XcpMaxCto**
Label	Maximum CTO Length
Description	The maximum length of a CTO packet in bytes, specific to each individual connection. A CTO packet consists of: ► an Identification Field (the PID containing the CTO packet code) ► the Data Field, which contains the specific payload necessary for the different types of CTO packet. The length of the CTO varies according to the needs and the used XCP transport layer. Range: ► CAN: 8

	► CAN-FD: 8 .. (maximum data length of a CAN-FD frame) ► FlexRay: 8 .. (maximum data length of a FlexRay frame (`XcpMaxFlexMsgLengthInit`) - FlexRay header length) ► Ethernet: 8 .. (maximum data length of a Ethernet frame (`XcpMaxEthernetMsgLengthInit`) - Ethernet header length) ► CDD: 8 .. 255 **Optimization Effect:** ► **RAM reduction (code):** Selecting a smaller value for this parameter reduces the RAM consumption of the module code.
Multiplicity	1..1
Type	INTEGER
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpMaxDto**
Label	Maximum DTO Length
Description	The maximum length of a DTO packet in bytes, specific to each individual connection. A DTO packet consists of: ► an Identification Field, which varies depending upon the Identification Field type (`XcpIdentificationFieldType`) ► an optional Timestamp Field, which varies depending upon the Timestamp Field Type(`XcpTimestampType`) ► the Data Field, which contains the data for synchronous acquisition and stimulation. The length of the CTO varies according to the needs and the used XCP transport layer. Range: ► CAN: 8 ► CAN-FD: 8 .. (maximum data length of a CAN-FD frame) ► FlexRay: 8 .. (maximum data length of a FlexRay frame (`XcpMaxFlexMsgLengthInit`) - FlexRay header length)

	► Ethernet: 8 .. (maximum data length of a Ethernet frame (`XcpMaxEthernetMsgLengthInit`) - Ethernet header length) ► CDD: 8 .. (maximum data length of a CDD frame) **Optimization Effect:** ► **RAM reduction (code):** Selecting a smaller value for this parameter reduces the RAM consumption of the module code. ► **RAM reduction (config):** Selecting a smaller value for this parameter reduces the RAM consumption of the module config. ► **ROM reduction (config):** Selecting a smaller value for this parameter reduces the ROM consumption of the module config.
Multiplicity	1..1
Type	INTEGER
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpMaxBS**
Label	XcpMaxBS
Description	The maximum number of blocks in a DOWNLOAD block sequence. Includes the initial DOWNLOAD command (ASAM parameter: MAX_BS). Range: ► 1 .. 255 **Optimization Effect:** ► **RAM reduction (code):** Selecting a smaller value for this parameter reduces the RAM consumption of the module code.
Multiplicity	1..1
Type	INTEGER
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpMaxCtoPgm**
Label	XcpMaxCtoPgm
Description	Defines the maximum length of XCP command transfer objects (CTO) in bytes for flash programming (ASAM parameter: `MAX_CTO_PGM`)

	A CTO packet consists of: ► an Identification Field (the PID containing the CTO Packet code) ► the Data Field, which contains the specific payload necessary for the different types of CTO packet. The length of the CTO varies according to the needs and the used XCP transport layer. Range: ► CAN: 8 ► CAN-FD: 8 .. (maximum data length of a CAN-FD frame) ► FlexRay: 8 .. (maximum data length of a FlexRay frame (`XcpMaxFlexMsgLengthInit` or `XcpFlxBufMaxLenInitValue`) - FlexRay header length) Dependency on parameter(s): ► Flash Programming Support (`XcpPgmSupported`): this parameter is editable only if flash programming is enabled. Note: Some older Xcp masters ignore this parameter. If your master is affected, it is advised to configure this parameter to the same value as XcpMaxCto. **Optimization Effect:** ► **RAM reduction (code):** Selecting a smaller value for this parameter reduces the RAM consumption of the module code.
Multiplicity	1..1
Type	INTEGER
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpMaxBSPgm**
Label	XcpMaxBSPgm
Description	Defines the maximum allowed block size as the initial PROGRAM command plus the number of consecutive command packets (PROGRAM_NEXT) in a block sequence for flash programming (ASAM parameter: `MAX_BS_PGM`). Range: ► 1 .. 255

	Dependency on parameter(s): ► Flash Programming Support (`XcpPgmSupported`): this parameter is editable only if flash programming is enabled. ► Master Block Mode Support (`XcpMasterBlockModePgmSupported`): this parameter is editable if master block mode support for flash programming is enabled. **Optimization Effect:** ► **RAM reduction (code):** Selecting a smaller value for this parameter reduces the RAM consumption of the module code.
Multiplicity	1..1
Type	INTEGER
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	AUTOSAR_ECUC

Parameter Name	**XcpTimestampEnabled**
Label	Timestamp Enabled
Description	Enables or disables timestamps in DTO packets for the current connection. Note that the actual number of bytes that will be used for the timestamp field in DTO packets is given by XcpGeneral/XcpTimestampType. Dependency on other configuration parameters: If XcpGeneral/XcpTimestampType is set to NO_TIME_STAMP, then XcpTimestampEnabled cannot be TRUE.
Multiplicity	1..1
Type	BOOLEAN
Configuration class	**PreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

5.2.1.15. XcpConnectionInterfaceType

Containers included		
Container name	**Multiplicity**	**Description**
XcpConnectionOverCAN	1..1	

Containers included		
XcpConnectionOverCANFD	1..1	
XcpConnectionOverFlexRay	1..1	
XcpConnectionOverEthernet	1..1	
XcpConnectionOverCDD	1..1	

5.2.1.16. XcpConnectionOverCAN

Parameters included	
Parameter name	**Multiplicity**
XcpCtoSlavePduRef	1..1
XcpBroadcastPduRef	1..1
XcpConnectionCanIfCfgRef	1..1

Parameter Name	XcpCtoSlavePduRef
Label	CTO Slave PDU Reference (Rx PDU)
Description	Reference to the Xcp Rx PDU belonging to the selected connection. It is used by the master to send CMD packages. To configure this parameter, proceed as follows: 1. Configure the Rx PDU required for the CTO, using the PDU Information tab. Refer to the parameter dependency below. 2. Add the configured Rx PDU to the Xcp Rx PDUs to Connection mapping tab. 3. Select the PDU reference from the drop-down list. Dependency on parameter(s): ► Xcp Rx PDUs to Connection mapping (`XcpRxPduConnectionInfo`): At least one Xcp connection Rx PDU must be configured in the Xcp Rx PDUs to Connection mapping container. All PDUs configured will be listed in the drop-down list next to the `CTO Slave PDU Reference` parameter. The referenced PDU must support the receiving of CMD packages. ► The referenced node must belong to the same connection. ► XCP connection type (`XcpConnectionInterfaceType`): The selected value for this parameter must be `XcpConnectionOverCAN`. Otherwise this parameter is not used.

Multiplicity	1..1	
Type	REFERENCE	
Range	node:paths(../../XcpRxPduConnectionInfo/*)	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**XcpBroadcastPduRef**
Label	Master Broadcast PDU reference (Rx PDU)
Description	Reference to the Xcp Rx PDU belonging to the selected connection. It is used by the master to broadcast the GET_SLAVE_ID command. Note: On the referenced ID, only the GET_SLAVE_ID command will be accepted. All other packages received with this PDU will be ignored. To configure this parameter, proceed as follows: 1. Configure the Rx PDU required for the CTO, using the PDU Information tab. Refer to the parameter dependency below. 2. Add the configured Rx PDU to the Xcp Rx PDUs to Connection mapping tab. 3. Select the PDU reference from the drop-down list. Dependency on parameter(s): ► Xcp Rx PDUs to Connection mapping (`XcpRxPduConnectionInfo`): At least one Xcp connection Rx PDU must be configured in the Xcp Rx PDUs to Connection mapping container. All PDUs configured will be listed in the drop-down list next to the `CTO Slave PDU Reference` parameter. The referenced PDU must support the receiving of CMD packages. ► The referenced node must belong to the same connection. ► XCP connection type (`XcpConnectionInterfaceType`): The selected value for this parameter must be `XcpConnectionOverCAN`. Otherwise this parameter is not used. ► `XcpPduSupportForStimDtoType` must be disabled for the referenced Xcp PDU, if the referenced PDU is used only to receive the broadcasted GET_SLAVE_ID command.
Multiplicity	1..1
Type	REFERENCE
Range	node:paths(../../XcpRxPduConnectionInfo/*)

Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**XcpConnectionCanIfCfgRef**
Label	Reference to CanIf configuration
Description	Reference to the CanIf interface where the Source/Destination PDUs, configured for the selected connection, can be found. This reference is used to validate/generate PDUs information for the assigned PDUs for the selected connection over CAN. Dependency on parameter(s): ► XCP connection type (`XcpConnectionInterfaceType`): The selected value for this parameter must be `XcpConnectionOverCAN`. Otherwise this parameter is not used. ► A valid CanIf configuration must be available for selection.
Multiplicity	1..1
Type	REFERENCE
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

5.2.1.17. XcpConnectionOverCANFD

Parameters included	
Parameter name	**Multiplicity**
XcpCanFdCtoSlavePduRef	1..1
XcpCanFdBroadcastPduRef	1..1
XcpCanFdMaxDlcRequired	1..1
XcpCanFdMaxDlc	1..1
XcpCanFdFillValue	1..1
XcpConnectionCanFdCanIfCfgRef	1..1

Parameter Name	**XcpCanFdCtoSlavePduRef**
Label	CAN-FD CTO Slave PDU Reference (Rx PDU)
Description	Reference to the Xcp Rx PDU belonging to the selected connection. It is used by the master to send CMD packages.

	To configure this parameter, proceed as follows: 1. Configure the Rx PDU required for the CTO, using the PDU Information tab. Refer to the parameter dependency below. 2. Add the configured Rx PDU to the Xcp Rx PDUs to Connection mapping tab. 3. Select the PDU reference from the drop-down list Dependency on parameter(s): ► Xcp Rx PDUs to Connection mapping (`XcpRxPduConnectionInfo`): At least one Xcp connection Rx PDU must be configured in the Xcp Rx PDUs to Connection mapping container. All PDUs configured will be listed in the drop-down list next to the `CTO Slave PDU Reference` parameter. The referenced PDU must support the receiving of CMD packages. ► The referenced node must belong to the same connection. ► XCP connection type (`XcpConnectionInterfaceType`): The selected value for this parameter must be `XcpConnectionOverCANFD`. Otherwise this parameter is not used.
Multiplicity	1..1
Type	REFERENCE
Range	node:paths(../../XcpRxPduConnectionInfo/*)
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpCanFdBroadcastPduRef**
Label	Master Broadcast PDU reference (Rx PDU)
Description	Reference to the Xcp Rx PDU belonging to the selected connection. It is used by the master to broadcast the GET_SLAVE_ID command. Note: On the referenced ID, only the GET_SLAVE_ID command is accepted. All other packages received with this PDU are ignored. To configure this parameter, proceed as follows: 1. Configure the Rx PDU required for the CTO, using the PDU Information tab. Refer to the parameter dependency below. 2. Add the configured Rx PDU to the Xcp Rx PDUs to Connection mapping tab. 3. Select the PDU reference from the drop-down list

	Dependency on parameter(s): ► Xcp Rx PDUs to Connection mapping (`XcpRxPduConnectionInfo`): At least one Xcp connection Rx PDU must be configured in the Xcp Rx PDUs to Connection mapping container. All PDUs configured are listed in the drop-down list next to the `CTO Slave PDU Reference` parameter. The referenced PDU must support the receiving of CMD packages. ► The referenced node must belong to the same connection. ► XCP connection type (`XcpConnectionInterfaceType`): The selected value for this parameter must be `XcpConnectionOverCANFD`. Otherwise this parameter is not used. ► `XcpPduSupportForStimDtoType` must be disabled for the referenced Xcp PDU, if the referenced PDU is used only to receive the broadcasted GET_SLAVE_ID command.
Multiplicity	1..1
Type	REFERENCE
Range	node:paths(../../XcpRxPduConnectionInfo/*)
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpCanFdMaxDlcRequired**
Label	CANFD Max DLC required
Description	Defines if the length of the XCP packages is given by `XcpCanFdMaxDlc` or not. For CAN-FD, the next larger defined DLC is considered for the calculation if `XcpCanFdMaxDlcRequired` is disabled and the DLC of a frame does not match one of the defined DLC values. If `XcpCanFdMaxDlcRequired` is enabled, always the corresponding DLC = `XcpCanFdMaxDlc` parameter is used for the calculation. Dependency on other parameter(s): This parameter is available only if the following parameter is enabled: ► Xcp on CAN-FD (`XcpOnCanFDEnabled`)
Multiplicity	1..1
Type	BOOLEAN
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	XcpCanFdMaxDlc
Label	CAN-FD Max DLC
Description	Defines the value of maximum data length of a CAN-FD frame. The maximum data length of a CAN-FD message shall be one of the following values: 8, 12, 16, 20, 24, 32, 48, 64. Range: ► DLC values: 8, 12, 16, 20, 24, 32, 48, 64 Dependency on other parameter(s): This parameter is available only if the following parameters are enabled: ► Xcp on CAN-FD (`XcpOnCanFDEnabled`).
Multiplicity	1..1
Type	INTEGER
Default value	64
Range	8 12 16 20 24 32 48 64
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	XcpCanFdFillValue
Label	CAN-FD Fill Byte Value
Description	Defines the value used for the fill bytes of CAN-FD packages. This parameter should be used to define the value for fill bytes. For CAN-FD, fill bytes are added to the data until the expected package length is reached. Range: ► Fill bytes Value: 0 .. 255

	Dependency on other parameter(s): This parameter is available only if the following parameters are enabled: ► Xcp on CAN-FD (`XcpOnCanFDEnabled`).
Multiplicity	1..1
Type	INTEGER
Default value	255
Range	<=255 >=0
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpConnectionCanFdCanIfCfgRef**
Label	Reference CAN-FD to CanIf configuration
Description	Reference to the CanIf interface where the Source/Destination PDUs, configured for the selected connection, can be found. This reference is used to validate/generate PDUs information for the assigned PDUs for the selected connection over CAN-FD. Dependency on parameter(s): ► XCP connection type (`XcpConnectionInterfaceType`): The selected value for this parameter must be `XcpConnectionOverCANFD`. Otherwise this parameter is not used. ► A valid CanIf configuration must be available for selection.
Multiplicity	1..1
Type	REFERENCE
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

5.2.1.18. XcpConnectionOverFlexRay

Containers included		
Container name	**Multiplicity**	**Description**
XcpFlexrayBufferCfg	1..n	Contains the configuration options for FlexRay hardware buffers.

Containers included		
		At least two buffers need to be fully configured: one for receiving CMD packet types (i.e. commands sent by the Xcp master to the Xcp slave), and one for transmitting the RES_-ERR packet types (i.e. response to the commands). The fully configured buffers must have initial valid FlexRay frame parameters (see details in groups FLX_BUFCfg and LPDU_IDCfg) as the FrIf will not implicitly initialize the Flexray frames that are marked as reconfigurable.

Parameters included	
Parameter name	**Multiplicity**
XcpFlxNodeAddress	1..1
XcpFlxHeaderAlignment	1..1
XcpPackMultiMsgInOneFlexRayFrame	1..1
XcpSequenceCorrectionEnabled	1..1
XcpConnectionFrIfCfgRef	1..1
XcpMaxFlexMsgLengthInit	1..1

Parameter Name	XcpFlxNodeAddress
Label	Node Address for XCP (NAX)
Description	Defines the node address for XCP (NAX) that is used for this node. This value is written to the NAX field of the FlexRay XCP header when a message is sent to the master and is also expected in messages received from the master in the NAX field. When multiple FlexRay are configured, it is allowed to share the same NAX address between connections, or to have different NAX addresses. Range: ► 0 .. 255
Multiplicity	1..1
Type	INTEGER
Default value	0
Range	<=255 >=0

Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**XcpFlxHeaderAlignment**
Label	Alignment of the FlexRay XCP Header
Description	Defines the alignment of the FlexRay XCP header. Range: ► PACKET_ALIGNMENT_8: 8-bit alignment is used. ► PACKET_ALIGNMENT_16: 16-bit alignment is used. ► PACKET_ALIGNMENT_32: 32-bit alignment is used. **Optimization Effect:** ► **RAM reduction (code):** Selecting a smaller value for this parameter reduces the RAM consumption of the module code.
Multiplicity	1..1
Type	ENUMERATION
Default value	PACKET_ALIGNMENT_8
Range	PACKET_ALIGNMENT_8 PACKET_ALIGNMENT_16 PACKET_ALIGNMENT_32
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpPackMultiMsgInOneFlexRayFrame**
Label	Enable Multiple Xcp Msgs In One FlexRay Frame
Description	Enables the concatenation of multiple XCP messages in one frame for FlexRay communication.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpSequenceCorrectionEnabled**

Label	Enable Sequence Correctness	
Description	Enables the sequence correction. Each sent frame contains a counter that is incremented for each Xcp packet.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**XcpConnectionFrIfCfgRef**
Label	Reference to FrIf configuration
Description	Reference to the FrIf interface where the source/destination PDUs, configured for the selected connection, can be found. This reference is used to validate/generate PDU information for the assigned PDUs for the selected connection over FlexRay. Dependency on parameter(s): ► XCP connection type (`XcpConnectionInterfaceType`): The selected value for this parameter must be `XcpConnectionOverFlexRay`. Otherwise this parameter is not used. ► A valid FrIf configuration must be available for selection.
Multiplicity	1..1
Type	REFERENCE
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpMaxFlexMsgLengthInit**
Label	Max Flexray Msg Length Initial
Description	Defines the initial maximum data length of a FlexRay frame (*MAX_FLX_LEN_BUF_x_init*) that a specific slave is able to receive or transmit in a specific XCP-dedicated buffer. This parameter is meaningful only if FLX_ASSIGN command support (`XcpSupportForFlxTPCommands`) is disabled because in this case both the Rx and Tx buffers have the same size.

	Note that the current implementation does not allow you to change MAX_FLX_-LEN_BUF_x. So MAX_FLX_LEN_BUF_x is always equal to MAX_FLX_LEN_-BUF_x_init. Range: ► 8 .. 254 Dependency on other parameter(s): This parameter is available if all of the following conditions are fulfilled: ► Xcp on FlexRay (`XcpOnFlexRayEnabled`) is enabled. ► Multiple Xcp Msgs In One Xcp FlexRay Frame (`XcpPackMultiMsgInOne-FlexRayFrame`) is enabled. ► FlexRay Transport Layer Commands Support (`XcpSupportForFlxTP-Commands`) is disabled. Hint: This parameter should be high enough to accommodate at least 2 DTOs of maximum size.
Multiplicity	1..1
Type	INTEGER
Default value	254
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

5.2.1.19. XcpFlexrayBufferCfg

Containers included		
Container name	**Multiplicity**	**Description**
FLX_BUFCfg	1..1	**Label:** FLX_BUF Configuration FLX_BUF Configuration This configuration group configures the LPDU reference, i.e. actual Flexray hardware buffer.
LPDU_IDCfg	1..1	**Label:** LPDU_ID Configuration Defines the configuration of the FlexRay frame.

5.2.1.20. FLX_BUFCfg

Parameters included	
Parameter name	**Multiplicity**
XcpFrIfLPDURef	1..1
XcpFlxTxConnectionInfoRef	1..1
XcpFlxRxConnectionInfoRef	1..1
XcpFlxBufId	1..1
XcpFlxBufMaxLen	1..1
XcpFlxBufMaxLenInitValue	1..1

Parameter Name	**XcpFrIfLPDURef**	
Label	Flexray LPDU_ID Reference	
Description	Reference to the FrIf_LPDU that is mapped to the FlexRay hardware buffer.	
Multiplicity	1..1	
Type	REFERENCE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**XcpFlxTxConnectionInfoRef**	
Label	Flexray Tx Connection Info Reference	
Description	Reference to the Xcp Tx connection info that contains the Xcp packet types supported by this buffer.	
Multiplicity	1..1	
Type	REFERENCE	
Range	node:paths(../../../../../XcpTxPduConnectionInfo/*)	
Configuration class	**PreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**XcpFlxRxConnectionInfoRef**
Label	Flexray Rx Connection Info Reference
Description	Reference to the Xcp Rx connection info that contains the Xcp packet types supported by this buffer.
Multiplicity	1..1

Type	REFERENCE	
Range	node:paths(../../../../../XcpRxPduConnectionInfo/*)	
Configuration class	**PreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**XcpFlxBufId**	
Label	Xcp Flexray Buffer ID	
Description	FLX_BUF The buffer ID, which uniquely identifies an Xcp Flexray hardware buffer	
Multiplicity	1..1	
Type	INTEGER	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**XcpFlxBufMaxLen**	
Label	Xcp Flexray Maximum Buffer Length Properties	
Description	MAX_FLX_LEN_BUF_x_init properties. Range: ► `FIXED`: MAX_FLX_LEN_BUF_x_init has an initial value and cannot be changed. In that case, it equals MAX_FLX_LEN_BUF_x. ► `VARIABLE_INITIALIZED`: MAX_FLX_LEN_BUF_x_init can be changed by FLX_ASSIGN and, thus, becomes MAX_FLX_LEN_BUF_x.	
Multiplicity	1..1	
Type	ENUMERATION	
Default value	VARIABLE_INITIALIZED	
Range	FIXED VARIABLE_INITIALIZED	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**XcpFlxBufMaxLenInitValue**
Label	Xcp Flexray Maximum Buffer Length Initial Value

Description	MAX_FLX_LEN_BUF_x_init. This parameter defines the initial maximum data length of the FlexRay frame. Range: ► 8 .. 0xFE	
Multiplicity	1..1	
Type	INTEGER	
Default value	8	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

5.2.1.21. LPDU_IDCfg

Parameters included	
Parameter name	**Multiplicity**
XcpFlxSlotId	1..1
XcpFlxSlotIdInitValue	1..1
XcpFlxOffset	1..1
XcpFlxOffsetInitValue	1..1
XcpFlxCycleRepetition	1..1
XcpFlxCycleRepetitionInitValue	1..1
XcpFlxChannel	1..1
XcpFlxChannelInitValue	1..1

Parameter Name	**XcpFlxSlotId**
Label	Xcp Flexray Slot Id Properties
Description	FLX_SLOT_ID parameter properties Range: ► `FIXED`: FLX_SLOT_ID has an initial value and cannot be changed. ► `VARIABLE`: FLX_SLOT_ID does not have an initial value and can be changed. ► `VARIABLE_INITIALIZED`: FLX_SLOT_ID has an initial value and can be changed.

Multiplicity	1..1
Type	ENUMERATION
Default value	VARIABLE
Range	FIXED VARIABLE VARIABLE_INITIALIZED
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpFlxSlotIdInitValue**
Label	Xcp Flexray Slot Id Initial Value
Description	FLX_SLOT_ID parameter This parameter defines the FlexRay slot ID that the FrIf_LPdu shall be configured to. Range: ► 1 .. 2047
Multiplicity	1..1
Type	INTEGER
Default value	1
Configuration class	**PreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpFlxOffset**
Label	Xcp Flexray Cycle Offset Properties
Description	OFFSET parameter properties Range: ► `FIXED`: OFFSET has an initial value and cannot be changed. ► `VARIABLE`: OFFSET does not have an initial value and can be changed. ► `VARIABLE_INITIALIZED`: OFFSET has an initial value and can be changed.
Multiplicity	1..1
Type	ENUMERATION

Default value	VARIABLE	
Range	FIXED VARIABLE VARIABLE_INITIALIZED	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**XcpFlxOffsetInitValue**	
Label	Xcp Flexray Cycle Offset Initial Value	
Description	OFFSET parameter This parameter contains the FlexRay offset (base cycle) that is used to transmit this FlexRay frame. Range: ► 0 .. 63	
Multiplicity	1..1	
Type	INTEGER	
Default value	0	
Configuration class	**PreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**XcpFlxCycleRepetition**
Label	Xcp Flexray Cycle Repetition Properties
Description	CYCLE_REPETITION parameter properties Range: ► `FIXED`: CYCLE_REPETITION has an initial value and cannot be changed. ► `VARIABLE`: CYCLE_REPETITION does not have an initial value and can be changed. ► `VARIABLE_INITIALIZED`: CYCLE_REPETITION has an initial value and can be changed.
Multiplicity	1..1
Type	ENUMERATION
Default value	VARIABLE

Range	FIXED	
	VARIABLE	
	VARIABLE_INITIALIZED	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**XcpFlxCycleRepetitionInitValue**	
Label	Xcp Flexray Cycle Repetition Initial Value	
Description	CYCLE_REPETITION parameter This parameter contains the FlexRay cycle repetition that is used to transmit the FlexRay frame. Range: ► 0 .. 0xFF	
Multiplicity	1..1	
Type	INTEGER	
Default value	0	
Configuration class	**PreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**XcpFlxChannel**
Label	Xcp Flexray Channel Properties
Description	CHANNEL parameter properties Range: ► `FIXED`: CHANNEL has an initial value and cannot be changed. ► `VARIABLE`: CHANNEL does not have an initial value and can be changed. ► `VARIABLE_INITIALIZED`: CHANNEL has an initial value and can be changed.
Multiplicity	1..1
Type	ENUMERATION
Default value	VARIABLE
Range	FIXED
	VARIABLE

	VARIABLE_INITIALIZED	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**XcpFlxChannelInitValue**
Label	Xcp Flexray Channel initial value
Description	CHANNEL parameter This parameter contains the FlexRay channel that is used to transmit the FlexRay frame. Range: ► A: Channel A is used. ► B: Channel B is used.
Multiplicity	1..1
Type	ENUMERATION
Default value	A
Range	A B
Configuration class	**PreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

5.2.1.22. XcpConnectionOverEthernet

Parameters included	
Parameter name	**Multiplicity**
XcpAutoOpenSoCon	1..1
XcpPackMultiMsgInOneEthernetFrame	1..1
XcpMaxEthernetMsgLengthInit	1..1
XcpConnectionSoAdConfigRef	1..1

Parameter Name	**XcpAutoOpenSoCon**
Label	Automatic Open for the Socket Connection

Description	Enables the automatic opening of the Ethernet socket(s) during start-up. If enabled, `SoAd_OpenSoCon()` is called during `Xcp_Init()` for each configured socket. If UDP protocol is used, this parameter must be enabled. Note: Ensure that `SoAd_Init()` is called before `Xcp_Init()`. Dependency on parameter(s): ► XCP connection type (`XcpConnectionInterfaceType`): The selected value for this parameter must be `XcpConnectionOverEthernet`. Otherwise this parameter is not used.
Multiplicity	1..1
Type	BOOLEAN
Default value	true
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpPackMultiMsgInOneEthernetFrame**
Label	Enable Multiple Xcp Msgs In One Ethernet Frame
Description	Enables the concatenation of multiple XCP messages in one frame, for Ethernet communication. Note: The ASAM specifications specify the support for multiple PDUs in one frame only for UDP/IP. If the selected communication type is TCP/IP, ensure that the Xcp master implementation supports multiple packing also for TCP connections before enabling this feature.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpMaxEthernetMsgLengthInit**
Label	Max Eternet Msg Length Initial
Description	Defines the initial maximum data length of an Ethernet frame that a specific slave is able to receive or transmit in a frame. It is measured in bytes.

	The length of the message can be variable. However, the maximum length of an XCP on Ethernet message is limited to the Ethernet payload length limits in bytes. Range: ► 8 .. 1454 Dependency on other parameter(s): This parameter is available only if both of the following parameters are enabled: ► Xcp on Ethernet (`XcpOnEthernetEnabled`) ► Multiple Xcp Msgs In One Xcp Frame (`XcpPackMultiMsgInOneEthernetFrame`) The default value for this parameter is calculated to always be able to accommodate 1 maximum size CTO plus 5 maximum size DTOs. Hint: This parameter should be high enough to accommodate at least 2 maximum size DTOs.
Multiplicity	1..1
Type	INTEGER
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpConnectionSoAdConfigRef**
Label	Reference to SoAd configuration
Description	Reference to the SoAd interface where the source/destination PDUs, configured for the selected connection, can be found. This reference is used to validate/generate PDU information for the assigned PDUs for the selected connection over Ethernet. Dependency on parameter(s): ► XCP connection type (`XcpConnectionInterfaceType`): The value selected for this parameter must be `XcpConnectionOverEthernet`. Otherwise this parameter is not used. ► A valid SoAd configuration must be available for selection. ► One entry with the name 'Xcp', containing the upper layer configuration for the Xcp module in SoAd, must be added into the `SoAdBswModules` container.

	▶ SoAdIf must be enabled for the dedicated Xcp configuration entry in SoAdBswModules. ▶ SoAdTp must be disabled for the dedicated Xcp configuration entry in SoAdBswModules. ▶ SoAdIfTxConfirmation must be enabled for the dedicated Xcp configuration entry in SoAdBswModules. ▶ SoAdLocalIpAddrAssigmentChg must be disabled for the dedicated Xcp configuration entry in SoAdBswModules (refer to ASAM specifications). ▶ SoAdIfTriggerTransmit must be disabled for the dedicated Xcp configuration entry in SoAdBswModules (the trigger transmit API is not supported for Ethernet). ▶ SoAdSoConModeChg parameter, configured for the dedicated Xcp configuration entry in SoAdBswModules, must be: ▶ Enabled, if there is configured at least one Xcp connection over TCP/IP. ▶ Disabled, if there is no Xcp connection over TCP/IP configured.
Multiplicity	1..1
Type	REFERENCE
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

5.2.1.23. XcpConnectionOverCDD

Containers included		
Container name	**Multiplicity**	**Description**
XcpCddInformation	1..1	**Label:** CDD Information

Parameters included	
Parameter name	**Multiplicity**
XcpCddCtoSlavePduRef	1..1
XcpCddBroadcastPduRef	1..1

Parameter Name	**XcpCddCtoSlavePduRef**
Label	CTO Slave PDU Reference (Rx PDU)
Description	Reference to the Xcp Rx PDU belonging to the selected connection. It is used, by the master, to send CMD packages.

	To configure this parameter, proceed as follows: 1. Configure the Rx PDU required for the CTO, using the PDU Information tab. Refer to the parameter dependency below. 2. Add the configured Rx PDU to the Xcp Rx PDUs to Connection mapping tab. 3. Select the PDU reference from the drop-down list. Dependency on parameter(s): ► Xcp Rx PDUs to Connection mapping (`XcpRxPduConnectionInfo`): At least one Xcp connection Rx PDU must be configured in the Xcp Rx PDUs to Connection mapping container. All PDUs configured are listed in the drop-down list next to the `CTO Slave PDU Reference` parameter. The referenced PDU must support the receiving of CMD packages. ► The referenced node must belong to the same connection. ► XCP connection type (`XcpConnectionInterfaceType`): The selected value for this parameter must be `XcpConnectionOverCDD`. Otherwise this parameter is not used.
Multiplicity	1..1
Type	REFERENCE
Range	node:paths(../../XcpRxPduConnectionInfo/*)
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpCddBroadcastPduRef**
Label	Master Broadcast PDU reference (Rx PDU)
Description	Reference to the Xcp Rx PDU belonging to the selected connection. It is used, by the master, to broadcast the GET_SLAVE_ID command. Note: On the referenced ID, only the GET_SLAVE_ID command is accepted. All other packages received with this PDU are ignored. To configure this parameter, proceed as follows: 1. Configure the Rx PDU required for the CTO, using the PDU Information tab. Refer to the parameter dependency below. 2. Add the configured Rx PDU to the Xcp Rx PDUs to Connection mapping tab. 3. Select the PDU reference from the drop-down list.

	Dependency on parameter(s): ▶ Xcp Rx PDUs to Connection mapping (`XcpRxPduConnectionInfo`): At least one Xcp connection Rx PDU must be configured in the Xcp Rx PDUs to Connection mapping container. All PDUs configured are listed in the drop-down list next to the `CTO Slave PDU Reference` parameter. The referenced PDU must support the receiving of CMD packages. ▶ The referenced node must belong to the same connection. ▶ XCP connection type (`XcpConnectionInterfaceType`): The value selected for this parameter must be `XcpConnectionOverCDD`. Otherwise this parameter is not used. ▶ `XcpPduSupportForStimDtoType` must be disabled for the referenced Xcp PDU, if the referenced PDU is used only to receive the broadcasted GET_SLAVE_ID command.
Multiplicity	1..1
Type	REFERENCE
Range	node:paths(../../XcpRxPduConnectionInfo/*)
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

5.2.1.24. XcpCddInformation

Parameters included	
Parameter name	**Multiplicity**
XcpCddShortName	1..1
XcpCddHeaderFile	1..1
XcpCddTransmitFunctionName	1..1

Parameter Name	**XcpCddShortName**
Label	CDD Short Name
Description	Short name of the CDD module interacting with Xcp.
Multiplicity	1..1
Type	STRING
Default value	Cdd
Configuration class	**VariantPreCompile:** VariantPreCompile

Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpCddHeaderFile**	
Label	CDD Header File	
Description	Defines the header file for the CDD interface APIs.	
Multiplicity	1..1	
Type	STRING	
Default value	Cdd.h	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**XcpCddTransmitFunctionName**	
Label	CDD Transmit Function Name	
Description	Defines the name of Cdd_Transmit().	
Multiplicity	1..1	
Type	FUNCTION-NAME	
Default value	Cdd_Transmit	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

5.2.1.25. XcpA2LGenerationSupport

Parameters included	
Parameter name	**Multiplicity**
XcpSynchEdge	1..1
XcpSampleRate	1..1
XcpBtlCyclesCan	1..1
XcpBtlCyclesCanFd	1..1
XcpMaxBusLoad	1..1

Parameter Name	**XcpSynchEdge**
Label	Synchronization Edge

Description	This parameter corresponds to the A2L parameter SYNC_EDGE.
Multiplicity	1..1
Type	ENUMERATION
Default value	SINGLE
Range	SINGLE DUAL
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpSampleRate**
Label	Sample Rate
Description	This parameter corresponds to the A2L parameter SAMPLE_RATE.
Multiplicity	1..1
Type	ENUMERATION
Default value	SINGLE
Range	SINGLE TRIPLE
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpBtlCyclesCan**
Label	BTL Cycles of CAN
Description	This parameter corresponds to the A2L parameter BTL_CYCLES of CAN.
Multiplicity	1..1
Type	INTEGER
Default value	0
Range	<=255 >=0
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpBtlCyclesCanFd**

Label	BTL Cycles of CANFD
Description	This parameter corresponds to the A2L parameter BTL_CYCLES of CANFD.
Multiplicity	1..1
Type	INTEGER
Default value	0
Range	<=255 >=0
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpMaxBusLoad**
Label	Max Bus Load
Description	This parameter corresponds to the A2L parameter MAX_BUS_LOAD.
Multiplicity	1..1
Type	INTEGER
Default value	0
Range	<=4294967295 >=0
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

5.2.1.26. XcpTxPduConnectionInfo

Parameters included	
Parameter name	**Multiplicity**
XcpPduSupportForResErrCtoType	1..1
XcpPduSupportForEvServCtoType	1..1
XcpPduSupportForDaqDtoType	1..1
XcpPduSupportTxFromTxConfirmation	1..1
XcpDefaultStateDynamicTxPDU	0..1
XcpMapTxPdu2Connection	1..1
XcpCddLowerLayerTxPduId	1..1

Parameter Name	XcpPduSupportForResErrCtoType	
Label	Support for transmitting RES/ERR packages	
Description	Enables transmitting `RES/ERR CTO` packages via the selected Tx PDU buffer.	
Multiplicity	1..1	
Type	BOOLEAN	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	XcpPduSupportForEvServCtoType	
Label	Support for transmitting EV/SERV packages	
Description	Enables transmitting `EV/SERV CTO` packages via the selected Tx PDU buffer. Dependency on parameter(s): ► This parameter is configurable only if the support for event packages is enabled (refer to `XcpEventPacketEnabled` parameter).	
Multiplicity	1..1	
Type	BOOLEAN	
Configuration class	**PreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	XcpPduSupportForDaqDtoType	
Label	Support for transmitting DAQ packages	
Description	Enables transmitting `DAQ DTO` packages via the selected Tx PDU buffer. Dependency on parameter(s): ► This parameter is configurable only if the DAQ list processor is enabled (refer to `XcpDaqSupported` parameter).	
Multiplicity	1..1	
Type	BOOLEAN	
Configuration class	**PreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	XcpPduSupportTxFromTxConfirmation
Label	Support for transmission from TxConfirmation

Description	Support for transmission from Tx confirmation. If enabled, a new transmission for this PDU is triggered when TxConfirmation is executed. **Enabling this feature for several PDUs might impact the run-time by creating a big overhead in the system.** Dependency on parameter(s): none.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**XcpDefaultStateDynamicTxPDU**	
Label	State of dynamic TxPdu	
Description	If the use of this parameter is enabled for a PDU, the transmission on this PDU at runtime can be enabled/disabled. If this parameter is not used, the PDU is enabled and static at runtime. If this parameter is set, the default state of the dynamic PDU is enabled. Dependency on parameter(s): ► XcpPduSupportForResErrCtoType: If a PDU is used for transmitting RES/ERR packages, it cannot be configured as dynamic. ► XcpSupportDynamicComChannels: This parameter is only available if dynamic communication channels are used.	
Multiplicity	0..1	
Type	BOOLEAN	
Default value	true	
Configuration class	**PreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**XcpMapTxPdu2Connection**
Label	Reference to Xcp Tx PDU
Description	Maps Tx PDUs to an Xcp connection.
Multiplicity	1..1
Type	REFERENCE

Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**XcpCddLowerLayerTxPduId**	
Label	CDD lower layer Tx PduId	
Description	Defines the lower layer PDU identifier for a connection over CDD, which must be used by the Xcp module for Cdd_Transmit().	
Multiplicity	1..1	
Type	INTEGER	
Default value	0	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

5.2.1.27. XcpRxPduConnectionInfo

Parameters included	
Parameter name	**Multiplicity**
XcpRxLowerLayerHandleId	1..1
XcpPduSupportForCmdCtoType	1..1
XcpPduSupportForStimDtoType	1..1
XcpDefaultStateDynamicRxPDU	0..1
XcpPduSupportRxFromRxIndication	1..1
XcpMapRxPdu2Connection	1..1

Parameter Name	**XcpRxLowerLayerHandleId**	
Label	Xcp Lower Layer handle Id	
Description	Xcp lower layer handle ID that is used by the lower layer communication protocol as notification handle ID upon a PDU receive indication notification.	
Multiplicity	1..1	
Type	INTEGER	
Default value	0	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	XcpPduSupportForCmdCtoType	
Label	Support for receiving CMD packages	
Description	Enables receiving `CMD CTO` packages via the selected Rx PDU buffer.	
Multiplicity	1..1	
Type	BOOLEAN	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	XcpPduSupportForStimDtoType	
Label	Support for receiving STIM packages	
Description	Enables receiving `STIM DTO` packages via the selected Rx PDU buffer. Dependency on parameter(s): ► This parameter is configurable only if the STIM processor is enabled (refer to `XcpStimSupported` parameter).	
Multiplicity	1..1	
Type	BOOLEAN	
Configuration class	**PreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	XcpDefaultStateDynamicRxPDU
Label	State of dynamic RxPdu
Description	If the use of this parameter is enabled for a PDU, the reception on this PDU at runtime can be enabled/disabled. If this parameter is not used, the PDU is enabled and static at runtime. If this parameter is set, the default state of the dynamic PDU is enabled. Dependency on parameter(s): ► XcpPduSupportForCmdCtoType: If a PDU is used for receiving CMD packages, it cannot be configured as dynamic. ► XcpSupportDynamicComChannels: This parameter is only available if dynamic communication channels are used.
Multiplicity	0..1
Type	BOOLEAN
Default value	true

Configuration class	**PreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**XcpPduSupportRxFromRxIndication**
Label	Support for reception from RxIndication
Description	Support for reception from Rx indication. If enabled, a reception for this PDU is triggered when RxIndication is executed. **Enabling this feature for several PDUs might impact the run-time by creating a big overhead in the system.** Dependency on parameter(s): none.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpMapRxPdu2Connection**
Label	Reference to Xcp Rx PDU
Description	Maps Rx PDUs to an Xcp connection.
Multiplicity	1..1
Type	REFERENCE
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

5.2.1.28. XcpMemoryAccessArea

Containers included		
Container name	**Multiplicity**	**Description**
XcpMemoryAccessArea	0..n	Defines the configuration of a range of memory addresses available for reading. If a Read operation needs to be performed on a certain memory address, the memory range received in the request (`memoryAddress` .. `memoryAddress` + `memorySize`

Containers included		
		-1) is checked if it is inside the allowed memory ranges, by checking all the configured `XcpMemoryAreaStartAddress` .. `XcpMemoryAreaStartAddress + XcpMemoryAreaLength` intervals. If it is outside the allowed range, a Negative Response Code is transmitted if the request is a calibration request, or an error ILLEGAL_MEMORY_ACCESS to DEM or DET (depending on the configuration) is issued. Dependency on parameter(s): ► Xcp Memory Access Areas Support (`XcpMemoryAccessAreasSupport`) must be enabled. ► Xcp Memory Access Areas Callout (`XcpMemoryAccessAreasCallout`) must be disabled.

Parameters included	
Parameter name	**Multiplicity**
XcpMemoryAccessAreasSupport	1..1
XcpMemoryAccessAreasCallout	1..1

Parameter Name	**XcpMemoryAccessAreasSupport**	
Label	Xcp Memory Access Areas Support	
Description	Enables memory access areas. ► true: Memory access areas are enabled. Accesses are allowed only for the memory areas defined. If no memory area is defined, all memory accesses are rejected. ► false: Memory access areas are disabled. All memory accesses are accepted.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**XcpMemoryAccessAreasCallout**

Label	Xcp Memory Access Areas Callout
Description	Enables the memory access areas callout feature. ► true: The user-specific callout function `Xcp_ApplIsMemoryAreaAccessible()` is called in order to verify if the access to a specific memory address is allowed. ► false: The verification of memory accesses is done using the configuration defined in the Xcp Memory Access Areas list. Dependency on parameter(s): ► Xcp Memory Access Areas Support (`XcpMemoryAccessAreasSupport`) must be enabled. ► Xcp Bsw Distribution feature (`XcpBswDistribution`) must be disabled. Note: Refer to the module documentation for the prototype of the callout function `Xcp_ApplIsMemoryAreaAccessible()`.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**PreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

5.2.1.29. XcpMemoryAccessArea

Parameters included	
Parameter name	**Multiplicity**
XcpMemoryAreaStartAddress	1..1
XcpMemoryAreaLength	1..1
XcpMemoryAreaAcccessType	1..1
XcpMemoryAreaAcccessScope	1..1
XcpMemoryAccessAreaApplicationRef	1..1
XcpMemoryAccessAreaReadRAMCallout	0..1
XcpMemoryAccessAreaWriteRAMCallout	0..1

Parameter Name	**XcpMemoryAreaStartAddress**
Label	Memory Area Start Address

Description	Defines the start address of this memory area. Enter here a valid C-expression that evaluates to an address in the slave's memory, e.g. `0xEB15C001` for a literal address (recommended) or `&VariableName` for a variable address (not recommended - we accept this for testing purposes). Any literal address must point to a valid address in the slave's memory. If variable names are used, the Xcp module expects all required symbol declarations and definitions to be available after including the User Header File, which is specified by the configuration parameter XcpGeneral/XcpUserHeader.	
Multiplicity	1..1	
Type	LINKER-SYMBOL	
Default value	0	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**XcpMemoryAreaLength**	
Label	Memory Area Length	
Description	Defines the length of this memory area. Range: ► 0 .. 4294967294	
Multiplicity	1..1	
Type	INTEGER	
Default value	0	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**XcpMemoryAreaAcccessType**
Label	Xcp Memory Access Type
Description	Defines the type of the access that this memory area accepts. Range: ► READ_WRITE: accept all memory accesses. ► READ: accept only memory read accesses. ► WRITE: accept only memory write accesses.
Multiplicity	1..1

Type	ENUMERATION	
Default value	READ_WRITE	
Range	READ_WRITE READ WRITE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**XcpMemoryAreaAcccessScope**
Label	Xcp Memory Access Scope
Description	Defines the scope of the access that this memory area accepts. Range: ► ALL: accept all requests. ► CALIBRATION: accept only download/upload/modify bits/build checksum requests. ► DAQ_STIM: accept only DAQ lists requests (DAQ or STIM direction).
Multiplicity	1..1
Type	ENUMERATION
Default value	ALL
Range	ALL CALIBRATION DAQ_STIM
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpMemoryAccessAreaApplicationRef**
Label	Xcp Memory Access Area Application
Description	Defines the Os application the Xcp Memory Access Area instance is mapped to.
Multiplicity	1..1
Type	REFERENCE
Configuration class	**PreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	XcpMemoryAccessAreaReadRAMCallout	
Label	Custom Read from RAM Callout	
Description	The user defined Callout for reading from RAM memory, for the memory Area.	
Multiplicity	0..1	
Type	STRING	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	XcpMemoryAccessAreaWriteRAMCallout	
Label	Custom Write to RAM Callout	
Description	The user defined Callout for writing to RAM memory, for the memory Area.	
Multiplicity	0..1	
Type	STRING	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

5.2.1.30. RamOptimizations

Parameters included	
Parameter name	**Multiplicity**
XcpEnableDaqOptimization	1..1
XcpEnableBitOffset	1..1

Parameter Name	XcpEnableDaqOptimization
Label	Daq Optimization
Description	Enables RAM optimization if DAQ List Support is enabled. Can be used to reduce the RAM usage of DAQ lists when a large number of ODT entries is required. This optimization reduces the RAM consumption regardless of the value parameter XcpDaqConfigType has. For backwards compatibility w.r.t storing DAQ Lists in non-volatile memory this parameter must be set to FALSE. Otherwise, the Service Needs Assistant needs to be run again as the associated NVM_BLOCK_XCP_DAQ_LISTS NvM block

	is different than the one produced when XcpEnableDaqOptimization is set to FALSE. At the same time, the data layout inside the block is also different.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpEnableBitOffset**
Label	Enable Bit Offset
Description	Enables the bit offset functionality in the context of RAM optimizations provided by enabling parameter XcpEnableDaqOptimization. If this parameter is set to FALSE then the memory consumption is reduced at maximum by not storing any information related to bit offsets in RAM. In this case, commands WRITE_DAQ and WRITE_DAQ_MULTIPLE that specify bit offsets different than 0xFF are rejected and error response XCP_ERR_OUT_OF_RANGE is sent to the master. Setting this parameter to TRUE allows the use of bit offsets while still providing some RAM consumption decrease. Note that if XcpEnableDaqOptimization is set to FALSE then XcpEnableBitOffset cannot be used at all and bit offsets are enabled.
Multiplicity	1..1
Type	BOOLEAN
Default value	true
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

5.2.1.31. ReportToDem

Parameters included	
Parameter name	**Multiplicity**
XcpRetryFailedReportToDem	1..1
XcpRetryFailedDemDetErrorId	1..1

Parameters included	
XcpIllegalMemoryAccessReportToDem	1..1
XcpIllegalMemoryAccessDemDetErrorId	1..1
XcpPduBufferLockedReportToDem	1..1
XcpPduBufferLockedDemDetErrorId	1..1
XcpRespCTOQueueFullReportToDem	1..1
XcpRespCTOQueueFullDemDetErrorId	1..1
XcpPDUBufferFullReportToDem	1..1
XcpPDUBufferFullDemDetErrorId	1..1
XcpPDULostReportToDem	1..1
XcpPDULostDemDetErrorId	1..1
XcpStimulationDataLostReportToDem	1..1
XcpStimulationDataLostDemDetErrorId	1..1
XcpAcquisitionDataLostReportToDem	1..1
XcpAcquisitionDataLostDemDetErrorId	1..1

Parameter Name	XcpRetryFailedReportToDem	
Label	Xcp Retry Failed Reporting Type	
Description	Defines the handling of the production error: *XCP_E_RETRY_FAILED* ► `DEM`: All errors are reported to the Diagnostics Event Manager (Dem). ► `DET`: All errors are reported to the Development Error Tracer (Det) if enabled. ► `DISABLE`: Production errors are not reported at all.	
Multiplicity	1..1	
Type	ENUMERATION	
Default value	DEM	
Range	DEM DET DISABLE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	XcpRetryFailedDemDetErrorId

Label	Xcp Retry Failed DET ErrorId
Description	Defines the `ErrorId` of the Xcp Retry Failed error when it is reported to Det instead of Dem. Range: ► 30 .. 255
Multiplicity	1..1
Type	INTEGER
Default value	30
Range	<=255 >=30
Configuration class	**PreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpIllegalMemoryAccessReportToDem**
Label	Xcp Illegal Memory Access Reporting Type
Description	Defines the handling of the production error: *XCP_E_ILLEGAL_MEMORY_ACCESS* ► `DEM`: All errors are reported to the Diagnostics Event Manager (Dem). ► `DET`: All errors are reported to the Development Error Tracer (Det) if enabled. ► `DISABLE`: Production errors are not reported at all.
Multiplicity	1..1
Type	ENUMERATION
Default value	DISABLE
Range	DEM DET DISABLE
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpIllegalMemoryAccessDemDetErrorId**
Label	Xcp Illegal Memory Access DET ErrorId

Description	Defines the `ErrorId` of the Xcp Illegal Memory Access error when it is reported to Det instead of Dem. Range: ► 30 .. 255
Multiplicity	1..1
Type	INTEGER
Default value	30
Range	<=255 >=30
Configuration class	**PreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpPduBufferLockedReportToDem**
Label	Xcp Pdu Buffer Locked Reporting Type
Description	Defines the handling of the production error: *XCP_E_PDU_BUFFER_LOCKED* ► `DEM`: All errors are reported to the Diagnostics Event Manager (Dem). ► `DET`: All errors are reported to the Development Error Tracer (Det) if enabled. ► `DISABLE`: Production errors are not reported at all.
Multiplicity	1..1
Type	ENUMERATION
Default value	DISABLE
Range	DEM DET DISABLE
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpPduBufferLockedDemDetErrorId**
Label	Xcp Pdu Buffer Locked DET ErrorId
Description	Defines the `ErrorId` of the Xcp PDU Buffer Locked error when it is reported to Det instead of Dem. Range:

	► 30 .. 255
Multiplicity	1..1
Type	INTEGER
Default value	130
Range	<=255 >=30
Configuration class	**PreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpRespCTOQueueFullReportToDem**
Label	Xcp Response CTO Queue Full Reporting Type
Description	Defines the handling of the production error: *XCP_E_RESP_CTO_QUEUE_FULL* ► `DEM`: All errors are reported to the Diagnostics Event Manager (Dem). ► `DET`: All errors are reported to the Development Error Tracer (Det) if enabled. ► `DISABLE`: Production errors are not reported at all.
Multiplicity	1..1
Type	ENUMERATION
Default value	DISABLE
Range	DEM DET DISABLE
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpRespCTOQueueFullDemDetErrorId**
Label	Xcp Response CTO Queue Full ErrorId
Description	Defines the `ErrorId` of the Xcp Response CTO Queue Full error when it is reported to Det instead of Dem. Range: ► 30 .. 255
Multiplicity	1..1

Type	INTEGER	
Default value	133	
Range	<=255 >=30	
Configuration class	**PreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**XcpPDUBufferFullReportToDem**	
Label	Xcp PDU Buffer Full Reporting Type	
Description	Defines the handling of the production error: *XCP_E_PDU_BUFFER_FULL* ► `DEM`: All errors are reported to the Diagnostics Event Manager (Dem). ► `DET`: All errors are reported to the Development Error Tracer (Det) if enabled. ► `DISABLE`: Production errors are not reported at all.	
Multiplicity	1..1	
Type	ENUMERATION	
Default value	DISABLE	
Range	DEM DET DISABLE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**XcpPDUBufferFullDemDetErrorId**
Label	Xcp PDU Buffer Full ErrorId
Description	Defines the `ErrorId` of the Xcp PDU Buffer Full error when it is reported to Det instead of Dem. Range: ► 30 .. 255
Multiplicity	1..1
Type	INTEGER
Default value	129

Range	<=255	
	>=30	
Configuration class	**PreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**XcpPDULostReportToDem**
Label	Xcp PDU Lost Reporting Type
Description	Defines the handling of the production error: *XCP_E_PDU_LOST* ► `DEM`: All errors are reported to the Diagnostics Event Manager (Dem). ► `DET`: All errors are reported to the Development Error Tracer (Det) if enabled. ► `DISABLE`: Production errors are not reported at all.
Multiplicity	1..1
Type	ENUMERATION
Default value	DISABLE
Range	DEM DET DISABLE
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpPDULostDemDetErrorId**
Label	Xcp PDU Lost ErrorId
Description	Defines the `ErrorId` of the Xcp PDU Lost error when it is reported to Det instead of Dem. Range: ► 30 .. 255
Multiplicity	1..1
Type	INTEGER
Default value	131
Range	<=255 >=30

Configuration class	**PreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**XcpStimulationDataLostReportToDem**
Label	Xcp Stimulation Data Lost Reporting Type
Description	Defines the handling of the production error: *XCP_E_STIMULATION_DATA_LOST* ► `DEM`: All errors are reported to the Diagnostics Event Manager (Dem). ► `DET`: All errors are reported to the Development Error Tracer (Det) if enabled. ► `DISABLE`: Production errors are not reported at all.
Multiplicity	1..1
Type	ENUMERATION
Default value	DISABLE
Range	DEM DET DISABLE
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpStimulationDataLostDemDetErrorId**
Label	Xcp Stimulation Data Lost ErrorId
Description	Defines the `ErrorId` of the Xcp Stimulation Data Lost error when it is reported to Det instead of Dem. Range: ► 30 .. 255
Multiplicity	1..1
Type	INTEGER
Default value	135
Range	<=255 >=30
Configuration class	**PreCompile:** VariantPreCompile

Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpAcquisitionDataLostReportToDem**	
Label	Xcp Acquisition Data Lost Reporting Type	
Description	Defines the handling of the production error: *XCP_E_ACQUISITION_DATA_LOST* ► `DEM`: All errors are reported to the Diagnostics Event Manager (Dem). ► `DET`: All errors are reported to the Development Error Tracer (Det) if enabled. ► `DISABLE`: Production errors are not reported at all.	
Multiplicity	1..1	
Type	ENUMERATION	
Default value	DISABLE	
Range	DEM DET DISABLE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**XcpAcquisitionDataLostDemDetErrorId**	
Label	Xcp Acquisition Data Lost ErrorId	
Description	Defines the `ErrorId` of the Xcp Acquisition Data Lost error when it is reported to Det instead of Dem. Range: ► 30 .. 255	
Multiplicity	1..1	
Type	INTEGER	
Default value	135	
Range	<=255 >=30	
Configuration class	**PreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

5.2.1.32. XcpBswDistribution

Parameters included	
Parameter name	**Multiplicity**
XcpMemoryProxyBufferSize	1..1
XcpMasterApplicationRef	1..1
XcpMemProxyNumberOfRequestQueued	1..1

Parameter Name	**XcpMemoryProxyBufferSize**	
Label	Memory Proxy Buffer Size	
Description	Defines the size of the buffer the memory proxy uses.	
Multiplicity	1..1	
Type	INTEGER	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**XcpMasterApplicationRef**	
Label	Xcp Master Partition	
Description	Defines the Os application the master Xcp instance is mapped to.	
Multiplicity	1..1	
Type	REFERENCE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**XcpMemProxyNumberOfRequestQueued**	
Label	Memory Proxy Number Of Queued Events	
Description	Defines the number of events that can be queued for MemProxy request, while waiting for the current request to be processed. This number is used in generating sizes of internal data, so a great number will affect the RAM usage. Range: ► 1 .. 255	
Multiplicity	1..1	
Type	INTEGER	
Default value	1	
Configuration class	**VariantPreCompile:**	VariantPreCompile

Origin	Elektrobit Automotive GmbH

5.2.1.33. XcpGeneral

Parameters included	
Parameter name	**Multiplicity**
XcpDaqSupported	1..1
XcpStimSupported	1..1
XcpDaqConfigType	1..1
XcpDaqCount	1..1
XcpDevErrorDetect	1..1
XcpIdentificationFieldType	1..1
XcpMainFunctionPeriod	1..1
XcpMaxCto	1..1
XcpMaxDto	1..1
XcpMaxEventChannel	1..1
XcpMinDaq	1..1
XcpOdtCount	1..1
XcpOdtEntriesCount	1..1
XcpOdtEntrySizeDaq	1..1
XcpOdtEntrySizeStim	1..1
XcpOnCanEnabled	1..1
XcpOnCanFDEnabled	1..1
XcpOnEthernetEnabled	1..1
XcpOnFlexRayEnabled	1..1
XcpOnCddEnabled	1..1
XcpPrescalerSupported	1..1
XcpSuppressTxSupport	1..1
XcpTimestampTicks	1..1
XcpTimestampType	1..1
XcpTimestampUnit	1..1
XcpVersionInfoApi	1..1
XcpCounterRef	1..1

Parameters included	
XcpEventTriggerMainFunc	1..1
XcpProcessMultiEventMainFunc	1..1
XcpDynamicAreaSize	1..1
XcpDynamicMirrorAreaSize	1..1
XcpCalPagSupported	1..1
XcpCalPagStoreSupported	1..1
XcpSlaveBlockModeSupported	1..1
XcpPgmSupported	1..1
XcpMaxCtoPgm	1..1
XcpMasterBlockModePgmSupported	1..1
XcpMaxBSPgm	1..1
XcpMinSTPgm	1..1
XcpAddressGranularity	1..1
XcpGranularityOdtEntrySizeDaq	1..1
XcpGranularityOdtEntrySizeStim	1..1
XcpUserHeader	1..1
XcpMasterBlockModeSupported	1..1
XcpMinST	1..1
XcpMaxBS	1..1
XcpTimestampFixed	1..1
XcpUserTimestampSupported	1..1
XcpStoreDaqSupported	1..1
XcpStoreDaqNvMBlock	1..1
XcpTxRetryCount	1..1
XcpTxBusTimeout	1..1
XcpTxBusRetry	1..1
XcpGetIdType1Callout	1..1
XcpGetIdType1MaxLength	1..1
XcpASAMMC2Filename	0..1
XcpSeedAndKeyEnabled	1..1
XcpEventPacketEnabled	1..1

Parameters included	
XcpModifyBitsSupported	1..1
XcpWriteDaqMultipleSupported	1..1
XcpBuildChecksumSupport	1..1
XcpBuildChecksumType	1..1
XcpBuildChecksumMainFunctionSupport	1..1
XcpCRCMainFunctionPeriod	1..1
XcpAddressTranslationSupport	1..1
XcpSetMtaEndianness	1..1
XcpBlockWriteReadMemoryRAMMechanism	1..1
XcpWriteMemoryRAMCallback	1..1
XcpReadMemoryRAMCallback	1..1
XcpEnableXcpControlApi	1..1
XcpDefaultXcpModuleState	1..1
XcpSupportForOverloadMSB	1..1
XcpSupportForGetDaqId	1..1
XcpSupportForFlxTPCommands	1..1
XcpSupportDynamicComChannels	1..1
XcpSupportComMainFunctions	0..1
XcpSupportForPDURoutingTable	1..1
XcpServiceWrapper	1..1
XcpFlexRaySyncTimeout	0..1
XcpMulticastCommands	1..1
XcpMulticastToUnicastRetry	1..1
XcpSynchReceivedCallout	1..1

Parameter Name	XcpDaqSupported
Label	DAQ List Support
Description	Enables the support of DAQ lists. ► true: DAQ list support is enabled. ► false: DAQ list support is disabled. To configure DAQ lists, proceed as follows:

	1. Enable DAQ list support. 2. If you want to use configurable DAQ lists (i.e. lists that are configured by the Xcp master during runtime), choose via parameter `XcpDaqConfigType` if you need `DAQ_STATIC` or `DAQ_DYNAMIC` lists. 3. Set the other DAQ list-related parameters in the container `XcpGeneral` according to the needs of your configuration (i.e. `XcpMinDaq`, `XcpOdtEntrySizeDaq`, `XcpOdtEntrySizeStim` and - if you set `XcpDaqConfigType` to `DAQ_DYNAMIC` - `XcpDaqCount` and `XcpOdtCount`). 4. If you have preconfigured DAQ lists (i.e. parameter `XcpMinDaq` is greater than 0) or if you use static configurable DAQ lists (i.e. parameter `XcpDaqConfigType` is set to `DAQ_STATIC`), configure the DAQ lists in the `XcpDaqList` containers. **Optimization Effect:** ► **ROM reduction (code):** Disabling this parameter reduces the ROM consumption of the module code. ► **RAM reduction (code):** Disabling this parameter reduces the RAM consumption of the module code. ► **Execution time reduction (code):** Disabling this parameter reduces the execution time of the module code.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPreCompile:** VariantPreCompile

Parameter Name	**XcpStimSupported**
Label	STIM Direction Support
Description	Enables the support of DAQ lists with direction STIM. ► true: Support of DAQ lists with direction STIM is enabled. ► false: Support of DAQ lists with direction STIM is disabled. To configure DAQ lists with direction STIM, proceed as follows: 1. Enable STIM direction support. 2. Enable DAQ list support and configure the DAQ lists according to the description for parameter `XcpDaqSupported`. Dependency on parameter(s):

	► DAQ list support (`XcpDaqSupported`): STIM direction support is only enabled if DAQ list support is enabled. **Optimization Effect:** ► **ROM reduction (code):** Disabling this parameter reduces the ROM consumption of the module code. ► **RAM reduction (code):** Disabling this parameter reduces the RAM consumption of the module code. ► **Execution time reduction (code):** Disabling this parameter reduces the execution time of the module code.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**PreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpDaqConfigType**
Label	DAQ Configuration Type
Description	Defines the type of configurable DAQ lists. Range: ► `DAQ_STATIC`: DAQ lists are configured statically. ► `DAQ_DYNAMIC`: DAQ lists are configured dynamically. Dependency on parameter(s): ► DAQ list support (`XcpDaqSupported`): must be enabled in order for this parameter to be editable. **Optimization Effect:** ► **ROM reduction (code):** Setting this parameter to a value of DAQ_STATIC reduces the ROM consumption of the module code. ► **RAM reduction (code):** Setting this parameter to a value of DAQ_STATIC reduces the RAM consumption of the module code.
Multiplicity	1..1
Type	ENUMERATION
Default value	DAQ_STATIC

Range	DAQ_DYNAMIC	
	DAQ_STATIC	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**XcpDaqCount**
Label	Number of DAQ Lists - Dynamic Config
Description	Defines the number of DAQ lists for `DYNAMIC` configurable DAQ list configuration (ASAM parameter: `DAQ_COUNT`). Range: ► 0 .. 65535 Note: `XcpDaqCount` + `XcpMinDaq` is the total number of DAQ lists (`DYNAMIC` and `PREDEFINED`) if `XcpConfigType` is set to `DAQ_DYNAMIC`. Dependency on parameter(s): ► DAQ list support (`XcpDaqSupported`): must be enabled in order for this parameter to be editable. ► DAQ Config Type (`XcpDaqConfigType`): this parameter is only used if configurable DAQ lists are `DYNAMIC`. **Optimization Effect:** ► **RAM reduction (config):** Selecting a smaller value for this parameter reduces the RAM consumption of the module configuration. ► **ROM reduction (config):** Selecting a smaller value for this parameter reduces the ROM consumption of the module configuration. ► **Execution time reduction (code):** Selecting a smaller value for this parameter reduces the execution time of the module code.
Multiplicity	1..1
Type	INTEGER
Default value	0
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	AUTOSAR_ECUC

Parameter Name	**XcpDevErrorDetect**
Label	Development Error Detection

Description	Enables the detection of development errors during development. **Optimization Effect:** ► **ROM reduction (code):** Disabling this parameter reduces the ROM consumption of the module code. ► **Execution time reduction(code):** Disabling this parameter reduces the execution time of the module code.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	true	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**XcpIdentificationFieldType**
Label	Identification Field Type
Description	Defines the identification field type for data transfer objects (DTOs). Range: ► `ABSOLUTE`: Absolute ODT number, i.e. the ODTs of all DAQ lists shall have one common numbering area that is zero-based and consecutive. ► `RELATIVE_BYTE`: Relative ODT number, absolute DAQ list number (BYTE), i.e. the ODTs of each DAQ list have continuous numbers starting with 0 for each DAQ list. The DAQ list numbers are also zero-based and consecutive and represented as 1-byte value. ► `RELATIVE_WORD`: Relative ODT number, absolute DAQ list number (WORD), i.e. the ODTs of each DAQ list have continuous numbers starting with 0 for each DAQ list. The DAQ list numbers are also zero-based and consecutive and represented as 2-byte value. ► `RELATIVE_WORD_ALIGNED`: Relative ODT number, absolute DAQ list number (WORD, aligned), i.e. the same as `RELATIVE_WORD` but the DAQ list number is word-aligned in the DTO. Dependency on parameter(s): ► DAQ list support (`XcpDaqSupported`): must be enabled in order for this parameter to be editable.
Multiplicity	1..1
Type	ENUMERATION

Default value	ABSOLUTE	
Range	ABSOLUTE	
	RELATIVE_BYTE	
	RELATIVE_WORD	
	RELATIVE_WORD_ALIGNED	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**XcpMainFunctionPeriod**
Label	Main Function Period [s]
Description	Defines the time interval in which the Xcp main function is invoked by the BSW Scheduler. Range: ► 0.001 .. 255
Multiplicity	1..1
Type	FLOAT
Default value	0.005
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	AUTOSAR_ECUC

Parameter Name	**XcpMaxCto**
Label	Maximum CTO Length
Description	**This parameter is only used as the default value of the connection-specific XcpMaxCto.** Defines the maximum length of XCP command transfer objects (CTO) in bytes (ASAM parameter: `MAX_CTO`). A CTO packet consists of: ► an Identification Field (the PID containing the CTO Packet code) ► the Data Field, which contains the specific payload necessary for the different types of CTO packet. The length of the CTO varies according to the needs and the used XCP transport layer.

	Range: ► CAN: 8 ► CAN-FD: 8 .. (maximum data length of a CAN-FD frame) ► FlexRay: 8 .. (maximum data length of a FlexRay frame (`XcpMaxFlexMsgLengthInit`) - FlexRay header length) ► Ethernet: 8 .. (maximum data length of a Ethernet frame (`XcpMaxEthernetMsgLengthInit`) - Ethernet header length) ► CDD: 8 .. 255 **Optimization Effect:** ► **RAM reduction (code):** Selecting a smaller value for this parameter reduces the RAM consumption of the module code.
Multiplicity	1..1
Type	INTEGER
Default value	8
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	AUTOSAR_ECUC

Parameter Name	**XcpMaxDto**
Label	Maximum DTO Length
Description	**This parameter is only used as the default value of the connection-specific XcpMaxDto.** Defines the maximum length of XCP Data Transfer Objects (DTO) in bytes (ASAM parameter: `MAX_DTO`). A DTO packet consists of: ► an Identification Field, which varies depending upon the Identification Field Type (`XcpIdentificationFieldType`) ► an optional Timestamp Field, which varies depending upon the Timestamp Field Type(`XcpTimestampType`) ► the Data Field, which contains the data for synchronous acquisition and stimulation. The length of the CTO varies according to the needs and the used XCP transport layer. Range:

	▶ CAN: 8 ▶ CAN-FD: 8 .. (maximum data length of a CAN-FD frame) ▶ FlexRay: 8 .. (maximum data length of a FlexRay frame (`XcpMaxFlexMsgLengthInit`) - FlexRay header length) ▶ Ethernet: 8 .. (maximum data length of a Ethernet frame (`XcpMaxEthernetMsgLengthInit`) - Ethernet header length) ▶ CDD: 8 .. (maximum data length of a CDD frame) **Optimization Effect:** ▶ **RAM reduction (code):** Selecting a smaller value for this parameter reduces the RAM consumption of the module code. ▶ **RAM reduction (config):** Selecting a smaller value for this parameter reduces the RAM consumption of the module config. ▶ **ROM reduction (config):** Selecting a smaller value for this parameter reduces the ROM consumption of the module config.
Multiplicity	1..1
Type	INTEGER
Default value	8
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	AUTOSAR_ECUC

Parameter Name	**XcpMaxEventChannel**
Label	Number of Event Channels
Description	Defines the number of configured events for DAQ lists (ASAM parameter: `MAX_EVENT_CHANNEL`). Note: This parameter is not editable because the number of event channels is used directly by counting the configured event channels (number of `XcpEventChannel`).
Multiplicity	1..1
Type	INTEGER
Default value	0
Range	<=65535 >=0
Configuration class	**VariantPreCompile:** VariantPreCompile

Origin	AUTOSAR_ECUC

Parameter Name	**XcpMinDaq**	
Label	Number of Predefined DAQ Lists	
Description	Defines the number of predefined DAQ lists (ASAM parameter: `MIN_DAQ`). Range: ► 0 .. 255 Dependency on parameter(s): ► DAQ list support (`XcpDaqSupported`): must be enabled in order for this parameter to be editable. **Optimization Effect:** ► **RAM reduction (config):** Selecting a smaller value for this parameter reduces the RAM consumption of the module configuration. ► **ROM reduction (code):** Selecting a smaller value for this parameter reduces the ROM consumption of the module code. ► **ROM reduction (config):** Selecting a smaller value for this parameter reduces the ROM consumption of the module configuration. ► **Execution time reduction (code):** Selecting a smaller value for this parameter reduces the execution time of the module code.	
Multiplicity	1..1	
Type	INTEGER	
Default value	0	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**XcpOdtCount**
Label	Maximum number of ODTs in each dynamic DAQ list
Description	Defines the maximum number of object descriptor tables (ODTs) allowed for each `DYNAMIC` DAQ list. Range: ► 0 .. 252 Dependency on parameter(s):

	► DAQ Config Type (`XcpDaqConfigType`): this parameter is only used if configurable DAQ lists are `DYNAMIC`.
	► DAQ list support (`XcpDaqSupported`): must be enabled in order for this parameter to be editable.
	► When the `XcpSupportForOverloadMSB` is enabled, not knowing the actual type of the DAQ list at run-time, the decreasing of the ODT range upper limit to '123' must be considered if the direction becomes 'DAQ' at run-time. Therefore, a warning is issued when this limit is exceeded. The following aspects are considered when the ODT range upper limit is calculated:
	  ► If the identification field is 'ABSOLUTE', see `XcpIdentificationFieldType`, all ODTs configured for the DAQ list with a lower ID than the selected one are considered. Also, all the ODTs with a lower ID than the selected one are considered, too, in the computation formula.
	  ► If the identification field is 'RELATIVE[/_-]', see `XcpIdentificationFieldType`, only the ODTs with a lower ID than the selected one are considered in the computation formula.
	► When the `XcpSupportForOverloadMSB` is disabled, not knowing the actual type of the DAQ list at run-time, the decreasing of the ODT range upper limit to '191' must be considered if the direction becomes 'STIM' at run-time. Therefore, a warning is issued when this limit is exceeded. The following aspects are considered when the ODT range upper limit is calculated:
	  ► If the identification field is 'ABSOLUTE', see `XcpIdentificationFieldType`, all ODTs configured for the DAQ list with a lower ID than the selected one are considered. Also, all the ODTs with a lower ID than the selected one are considered, too, in the computation formula.
	  ► If the identification field is 'RELATIVE[/_-]', see `XcpIdentificationFieldType`, only the ODTs with a lower ID than the selected one are considered in the computation formula.
	Optimization Effect:
	► **ROM reduction (config):** Selecting a smaller value for this parameter reduces the ROM consumption of the module configuration.
	► **RAM reduction (config):** Selecting a smaller value for this parameter reduces the RAM consumption of the module configuration.
	► **Execution time reduction (code):** Selecting a smaller value for this parameter reduces the execution time of the module code.

Multiplicity	1..1
Type	INTEGER

Default value	0	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**XcpOdtEntriesCount**
Label	Maximum number of ODT Entries in each ODT of a dynamic DAQ list
Description	Defines the maximum number of ODT entries into an object descriptor table (ODT) belonging to a dynamic DAQ list. Range: ► 1 .. 255 Dependency on parameter(s): ► DAQ list support (`XcpDaqSupported`): must be enabled in order for this parameter to be editable. ► DAQ Config Type (`XcpDaqConfigType`): this parameter is only used if configurable DAQ lists are `DYNAMIC`. **Optimization Effect:** ► **RAM reduction (config):** Selecting a smaller value for this parameter reduces the RAM consumption of the module configuration. ► **ROM reduction (config):** Selecting a smaller value for this parameter reduces the ROM consumption of the module configuration. ► **Execution time reduction (code):** Selecting a smaller value for this parameter reduces the execution time of the module code.
Multiplicity	1..1
Type	INTEGER
Default value	0
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	AUTOSAR_ECUC

Parameter Name	**XcpOdtEntrySizeDaq**
Label	Maximum Size for ODT Entry - DAQ
Description	Defines the maximum size of a DAQ list ODT entry (ASAM parameter: `MAX_ODT_ENTRY_SIZE_DAQ`). Range:

	► 0 .. 255 Dependency on parameter(s): ► DAQ list support (`XcpDaqSupported`): must be enabled in order for this parameter to be editable. Note: This parameter is counted in address granularity (AG, parameter `XcpAddressGranularity`) bytes.
Multiplicity	1..1
Type	INTEGER
Default value	1
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	AUTOSAR_ECUC

Parameter Name	**XcpOdtEntrySizeStim**
Label	Maximum Size for ODT Entry - STIM
Description	Defines the maximum size of an ODT entry (ASAM parameter: `MAX_ODT_ENTRY_SIZE_STIM`) for DAQ lists with direction STIM. Range: ► 0 .. 255 Dependency on parameter(s): ► DAQ list support (`XcpDaqSupported`): must be enabled in order for this parameter to be editable. Note: This parameter is counted in address granularity (AG, parameter `XcpAddressGranularity`) bytes.
Multiplicity	1..1
Type	INTEGER
Default value	1
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	AUTOSAR_ECUC

Parameter Name	**XcpOnCanEnabled**
Label	Enable XCP on CAN
Description	Enables XCP on a CAN network.

	Dependency on parameter(s): ► At least one connection of `XcpConnectionOverCAN` must be configured, if Xcp on CAN is enabled (refer to `XcpConnectionInterfaceType`). ► When Xcp on CAN is enabled, the `CanIf` module is required.
Multiplicity	1..1
Type	BOOLEAN
Default value	true
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	AUTOSAR_ECUC

Parameter Name	**XcpOnCanFDEnabled**
Label	Enable XCP on CAN-FD
Description	Enables XCP on a CAN-FD network. Dependency on parameter(s): ► At least one connection of `XcpConnectionOverCANFD` must be configured, if Xcp on CAN-FD is enabled (refer to `XcpConnectionInterfaceType`). ► When Xcp on CAN-FD is enabled, the `CanIf` module is required.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpOnEthernetEnabled**
Label	Enable XCP on Ethernet
Description	Enables XCP on a Ethernet network. Dependency on parameter(s): ► At least one connection of `XcpConnectionOverEthernet` must be configured, if Xcp on Ethernet is enabled (refer to `XcpConnectionInterfaceType`). ► When Xcp on Ethernet is enabled, the `SoAd` module is required.
Multiplicity	1..1

Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**XcpOnFlexRayEnabled**	
Label	Enable XCP on FlexRay	
Description	Enables XCP on a FlexRay network. Dependency on parameter(s): ► At least one connection of `XcpConnectionOverFlexRay` must be configured, if Xcp on FlexRay is enabled (refer to `XcpConnectionInterfaceType`). ► When Xcp on FlexRay is enabled, the `FrIf` module is required.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**XcpOnCddEnabled**	
Label	Enable XCP on CDD	
Description	Enables XCP on CDD (Complex Device Driver). Dependency on parameter(s): ► At least one connection of `XcpConnectionOverCDD` must be configured, if Xcp on CDD is enabled (refer to `XcpConnectionInterfaceType`). ► When Xcp on CDD is enabled, a `BusIf` module for the CDD is required.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**XcpPrescalerSupported**

Label	Prescaler Supported
Description	Enables the usage of the prescaler in DAQ lists for reducing the transmission period. Dependency on parameter(s): ► DAQ list support (`XcpDaqSupported`): must be enabled in order for this parameter to be editable. **Optimization Effect:** ► **ROM reduction (code):** Disabling this parameter reduces the ROM consumption of the module code. ► **Execution time reduction (code):** Disabling this parameter reduces the execution time of the module code.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	AUTOSAR_ECUC

Parameter Name	**XcpSuppressTxSupport**
Label	Transmit Suppression
Description	Enables the support of suppressing transmission of PDUs per communication channel on or off (via `Xcp_SetTransmissionMode()` API). **Optimization Effect:** ► **ROM reduction (code):** Disabling this parameter reduces the ROM consumption of the module code. ► **RAM reduction (code):** Disabling this parameter reduces the RAM consumption of the module code.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	AUTOSAR_ECUC

Parameter Name	**XcpTimestampTicks**

Label	Timestamp Ticks
Description	Defines the timestamp ticks. If User Timestamp Support (`XcpUserTimestampSupported`) is disabled, the calculation of the timestamp ticks is automatically generated, based upon the value `OsSecondsPerTick` of the OS Counter referred by `XcpCounterRef`. The timestamp ticks are calculated as smallest integer number to be multiplied with a power of ten to represent `OsSecondsPerTick` . Example: OsSecondsPerTick = 0.000625 = 625E-6 => Timestamp ticks is 625 (6250 would not be the smallest such number and 62.5 is no integer number at all). Dependency on parameter(s): ► DAQ list support (`XcpDaqSupported`): must be enabled in order for this parameter to be editable. ► Timestamp Type (`XcpTimestampType`): must be different than 'NO_-TIME_STAMP' in order for this parameter to be editable. ► User Timestamp Support (`XcpUserTimestampSupported`): must be 'true' in order for this parameter to be editable. ► OS Counter (`XcpCounterRef`): must be valid and have `OsSe-condsPerTick` configured if User Timestamp Support `XcpUserTime-stampSupported` is 'false'.
Multiplicity	1..1
Type	INTEGER
Default value	0
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	AUTOSAR_ECUC

Parameter Name	**XcpTimestampType**
Label	Timestamp Type
Description	Defines the number of bytes required for timestamp. Range: ► `NO_TIME_STAMP`: Timestamp shall not be used. ► `ONE_BYTE`: Timestamp shall be of 1 BYTE. ► `TWO_BYTE`: Timestamp shall be of 2 BYTE.

	► FOUR_BYTE: Timestamp shall be of 4 BYTE. Dependency on parameter(s): ► DAQ list support (XcpDaqSupported): must be enabled in order for this parameter to be editable. **Optimization Effect:** ► **ROM reduction (code):** Setting this parameter to a value of NO_TIME_STAMP reduces the ROM consumption of the module code. ► **Execution time reduction (code):** Selecting a smaller value for this parameter reduces the execution time of the module code.
Multiplicity	1..1
Type	ENUMERATION
Default value	NO_TIME_STAMP
Range	FOUR_BYTE NO_TIME_STAMP ONE_BYTE TWO_BYTE
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	AUTOSAR_ECUC

Parameter Name	**XcpTimestampUnit**
Label	Timestamp Unit
Description	Defines the timestamp unit. If User Timestamp Support XcpUserTimestampSupported is disabled, the calculation of the timestamp unit is automatically generated, based upon the value OsSecondsPerTick of the OS Counter referred by XcpCounterRef. The timestamp unit is chosen as such, that when combined with the timestamp ticks XcpTimestampTicks, the timestamp matches the value OsSecondsPerTick. Example (continued from example at XcpTimestampTicks): OsSecondsPerTick = 0.000625 = 625E-6 => Timestamp ticks is 625 and timestamp unit is 1 microsecond. Dependency on parameter(s):

	▶ DAQ list support (`XcpDaqSupported`): must be enabled in order for this parameter to be editable. ▶ Timestamp Type (`XcpTimestampType`): must be different than 'NO_-TIME_STAMP' in order for this parameter to be editable. ▶ User Timestamp Support (`XcpUserTimestampSupported`): must be 'true' in order for this parameter to be editable. ▶ OS Counter (`XcpCounterRef`): must be valid and have `OsSe-condsPerTick` configured if User Timestamp Support `XcpUserTime-stampSupported` is 'false'.
Multiplicity	1..1
Type	ENUMERATION
Default value	TIMESTAMP_UNIT_1S
Range	TIMESTAMP_UNIT_100MS TIMESTAMP_UNIT_100NS TIMESTAMP_UNIT_100PS TIMESTAMP_UNIT_100US TIMESTAMP_UNIT_10MS TIMESTAMP_UNIT_10NS TIMESTAMP_UNIT_10PS TIMESTAMP_UNIT_10US TIMESTAMP_UNIT_1MS TIMESTAMP_UNIT_1NS TIMESTAMP_UNIT_1PS TIMESTAMP_UNIT_1S TIMESTAMP_UNIT_1US
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	AUTOSAR_ECUC

Parameter Name	**XcpVersionInfoApi**
Label	Version Info API
Description	Enables the version information API (`Xcp_GetVersionInfo()`) **Optimization Effect:** ▶ **ROM reduction (code):** Disabling this parameter reduces the ROM consumption of the module code.

Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**XcpCounterRef**	
Label	OS Counter Reference	
Description	Reference to the counter that is used by Xcp to provide the timestamp. Dependency on parameter(s): ► DAQ list support (`XcpDaqSupported`): must be enabled in order for this parameter to be editable. ► Os Seconds Per Tick (`OsSecondsPerTick`): The Os Seconds Per Tick parameter of the referenced Os counter must be enabled so that timestamp can be correctly calculated. ► User Timestamp Support (`XcpUserTimestampSupported`): must be set to "false" in order for this parameter to be editable. ► Timestamp Type (`XcpTimestampType`): must be different than "NO_TIME_STAMP" in order for this parameter to be editable.	
Multiplicity	1..1	
Type	REFERENCE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**XcpEventTriggerMainFunc**
Label	Xcp Event Triggered Main Function
Description	Enables separately Event Triggered Main Function. ► true: Enables separately Event Main Function. ► false: Disable separately Event Main Function. Dependency on parameter(s): ► `XcpDaqSupported` must be enabled in order to have the "Xcp Event Triggered Main Function" configurable.

	► `XcpEventChannel` must have at least one child to allow having the "Xcp Event Triggered Main Function" feature configurable, when at least one cyclic event channel has to be configured.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**PreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpProcessMultiEventMainFunc**
Label	Xcp Process Multiple Events Per Main Function Cycle
Description	Enables the processing of multiple events per Xcp_MainFunction. ► true: Xcp processes all events that are assigned on Xcp_MainFunction, on each cycle. ► false: Xcp processes only one event on each Xcp_MainFunction cycle. Dependency on parameter(s): ► `XcpDaqSupported` must be enabled in order to have the "Xcp Process Multiple Events Per Main Function Cycle" configurable. ► `XcpEventChannel` must have at least one child to allow having the "Xcp Process Multiple Events Per Main Function Cycle" feature configurable, in the following situations: ► at least one sporadic event channel must be configured, or ► `XcpEventTriggerMainFunc` is disabled, if only cyclical event channels are configured.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**PreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpDynamicAreaSize**
Label	Allocated Size for Dynamic DAQ lists
Description	The size, in bytes, of the memory area used for dynamic DAQ list configuration.

	Note: To give a hint on how much size a project would need, here is the consumption of DAQ lists for the Windows platform, using the GCC compiler. Keep in mind that these values can vary, depending on the hardware and compiler, but the magnitude should be similar: ► 1 DAQ list: 16 bytes ► 1 ODT: 8 bytes ► 1 ODT entry: 8 bytes Range: ► 1 .. 4294967292 Dependency on parameter(s): ► DAQ list support (`XcpDaqSupported`): must be enabled in order to make this parameter editable. ► DAQ Config Type (`XcpDaqConfigType`): this parameter is only used if configurable DAQ lists are `DAQ_DYNAMIC`. **Optimization Effect:** ► **RAM reduction (config):** Selecting a smaller value for this parameter reduces the RAM consumption of the module configuration.
Multiplicity	1..1
Type	INTEGER
Default value	1024
Configuration class	**PreCompile:** VariantPreCompile

Parameter Name	**XcpDynamicMirrorAreaSize**
Label	Allocated Mirror Size for Dynamic DAQ lists
Description	The size, in bytes, of the memory area used for storing a mirror of the data from all configured ODT from a dynamic DAQ list. Note: To give a hint on how much size a project would need, here is the memory needed to store data for all configured ODT. It is equal to: ► XcpDaqCount*XcpOdtCount*XcpOdtEntriesCount*MaximOdtEntry *Granularity bytes ► MaximOdtEntry is the maximum between XcpOdtEntrySizeStim and XcpOdtEntrySizeDaq

	Range: ▶ 1 .. 4294967295 Dependency on parameter(s): ▶ DAQ list support (`XcpDaqSupported`): must be enabled in order for this parameter to be editable. ▶ DAQ Config Type (`XcpDaqConfigType`): this parameter is only used if configurable DAQ lists are `DAQ_DYNAMIC`. **Optimization Effect:** ▶ **RAM reduction (config):** Selecting a smaller value for this parameter reduces the RAM consumption of the module configuration.
Multiplicity	1..1
Type	INTEGER
Default value	1024
Configuration class	**PreCompile:** VariantPreCompile

Parameter Name	**XcpCalPagSupported**
Label	Calibration Page Support
Description	Enables the calibration page switching feature. This functionality allows you to control the slave's logical memory layout (segments, pages). ▶ true: The page switching commands are available. ▶ false: The page switching commands are not available. The following is the list of the calibration page switching commands together with the names of their callout functions. ▶ `SET_CAL_PAGE`. Callout function: `Xcp_ApplSetCalPage()` ▶ `GET_CAL_PAGE`. Callout function: `Xcp_ApplGetCalPage()` ▶ `GET_PAG_PROCESSOR_INFO`. Callout function: `Xcp_ApplGetPagProcessorInfo()` ▶ `GET_SEGMENT_INFO`. Callout function: `Xcp_ApplGetSegmentInfo()` ▶ `GET_PAGE_INFO`. Callout function: `Xcp_ApplGetPageInfo()` ▶ `SET_SEGMENT_MODE`. Callout function: `Xcp_ApplSetSegmentMode()` ▶ `GET_SEGMENT_MODE`. Callout function: `Xcp_ApplGetSegmentMode()` ▶ `COPY_CAL_PAGE`. Callout function: `Xcp_ApplCopyCalPage()`

	The slave must initialize all its PAGEs for all its SEGMENTs. To do this, an additional callout function is needed: ► SEGMENT/PAGE Initialization. Callout function: `Xcp_ApplCalPagInit()` The callout functions listed above must be implemented in the application. Note: Refer to the module documentation for the callout function prototypes. **Optimization Effect:** ► **ROM reduction (code):** Disabling this parameter reduces the ROM consumption of the module code.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpCalPagStoreSupported**
Label	Store Calibration Data Support
Description	Enables the storing of calibration data to non-volatile (NV) memory. This means that the `SET_REQUEST` command `STORE_CAL_REQUEST` mode is enabled. ► true: Storing of calibration data to NV memory feature is enabled. ► false: Storing of calibration data to NV memory feature is disabled. The `FREEZE_SUPPORTED` flag in `PAG_PROPERTIES` at `GET_PAG_PROCESSOR_INFO` indicates that all SEGMENTS can be put in FREEZE mode. With `SET_SEGMENT_MODE`, the master can select a SEGMENT for freezing. With the `STORE_CAL_REQ` mode parameter bit set in `SET_REQUEST` command, the master requests the slave to save calibration data to NV memory. This functionality is implemented via the callout function: `Xcp_ApplSetReqStoreCalReq()`. This callout function must be implemented in the application. When the store calibration data request is fulfilled, the `STORE_CAL_REQ` bit obtained by `GET_STATUS` must be reset and also an `EV_STORE_CAL` event packet may be sent by the slave. To do this, the application must call the callback function: `Xcp_SetReqStoreCalReqCbk()`. Dependency on parameter(s):

	► Calibration Page Switching (`XcpCalPagSupported`) must be enabled Note: Refer to the module documentation for the prototype of the callout function `Xcp_ApplSetReqStoreCalReq()` and the callback function `Xcp_SetReqStoreCalReqCbk()`. **Optimization Effect:** ► **ROM reduction (code):** Disabling this parameter reduces the ROM consumption of the module code.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**PreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpSlaveBlockModeSupported**
Label	Slave Block Mode Support
Description	Enables the slave block mode functionality. Affected command(s): ► UPLOAD **Optimization Effect:** ► **ROM reduction (code):** Disabling this parameter reduces the ROM consumption of the module code. ► **RAM reduction (code):** Disabling this parameter reduces the RAM consumption of the module code. ► **Execution time reduction (code):** Disabling this parameter reduces the execution time of the module code.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpPgmSupported**
Label	Flash Programming Support

Description	Enables the flash programming functionality. The following is the list of the flash programming commands together with the names of their callout functions. ► `PROGRAM_START`. Callout function: `Xcp_ApplProgramStart()` ► `PROGRAM_CLEAR`. Callout function: `Xcp_ApplProgramClear()` ► `PROGRAM/PROGRAM_NEXT/PROGRAM_MAX`. Callout function: `Xcp_ApplProgram()` ► `PROGRAM_RESET`. Callout function: `Xcp_ApplProgramReset()` ► `GET_PGM_PROCESSOR_INFO`. Callout function: `Xcp_ApplGetPgmProcessorInfo()` ► `GET_SECTOR_INFO`. Callout function: `Xcp_ApplGetSectorInfo()` The callout functions listed above must be implemented in the application. Note: Refer to the module documentation for the callout function prototypes. **Optimization Effect:** ► **ROM reduction (code):** Disabling this parameter reduces the ROM consumption of the module code.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpMaxCtoPgm**
Label	Max Cto Pgm
Description	**This parameter is only used as the default value of the connection specific XcpMaxCtoPgm.** Defines the maximum length of XCP command transfer objects (CTO) in bytes for flash programming (ASAM parameter: `MAX_CTO_PGM`). A CTO packet consists of: ► An Identification Field (the PID containing the CTO Packet code) ► And the Data Field which contains the specific payload necessary for the different types of CTO packet.

	The length of the CTO varies according to the needs and the used XCP Transport Layer. Range: ► CAN : 8 ► CAN-FD : 8..(maximum data length of a CAN-FD frame ► FlexRay : 8..(maximum data length of a FlexRay frame (`XcpMaxFlexMsgLengthInit` or `XcpFlxBufMaxLenInitValue`) - FlexRay header length) Dependency on parameter(s): ► Flash Programming Support (`XcpPgmSupported`): this parameter is editable only if flash programming is enabled. Note: Some older Xcp masters ignore this parameter. If your master is affected, it is advised to configure this parameter to the same value as XcpMaxCto. **Optimization Effect:** ► **RAM reduction (code):** Selecting a smaller value for this parameter reduces the RAM consumption of the module code.
Multiplicity	1..1
Type	INTEGER
Default value	8
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpMasterBlockModePgmSupported**
Label	Master Block Mode Pgm Supported
Description	Enables the master block mode functionality during programming. Affected command(s): ► PROGRAM ► PROGRAM_NEXT Dependency on parameter(s): ► Flash Programming Support (`XcpPgmSupported`): this parameter is editable only if flash programming is enabled.

	Optimization Effect: ► **ROM reduction (code):** Disabling this parameter reduces the ROM consumption of the module code.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**PreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpMaxBSPgm**
Label	Maximum Block Size for Flash Programming
Description	**This parameter is only used as the default value of the connection-specific XcpMaxBSPgm.** Defines the maximum allowed block size as the initial PROGRAM command plus the number of consecutive command packets (PROGRAM_NEXT) in a block sequence for flash programming (ASAM parameter: `MAX_BS_PGM`). Range: ► 1 .. 255 Dependency on parameter(s): ► Flash Programming Support (`XcpPgmSupported`): this parameter is editable only if flash programming is enabled. ► Master Block Mode Support (`XcpMasterBlockModePgmSupported`): this parameter is editable if master block mode support for flash programming is enabled. **Optimization Effect:** ► **RAM reduction (code):** Selecting a smaller value for this parameter reduces the RAM consumption of the module code.
Multiplicity	1..1
Type	INTEGER
Configuration class	**PreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpMinSTPgm**

Label	Minimum Separation Time for Flash Programming [100us]
Description	Defines the required minimum separation time between the flash programming packets of a block transfer from the master device to the slave device in units of 100 microseconds (ASAM parameter: `MIN_ST_PGM`). Range: ► 0 .. 255 Dependency on parameter(s): ► Flash Programming Support (`XcpPgmSupported`): this parameter is editable only if flash programming is enabled. ► Master Block Mode Support (`XcpMasterBlockModePgmSupported`): this parameter is editable if master block mode support for flash programming is enabled. ► Main Function Period (`XcpMainFunctionPeriod`): this parameter must be lower than or equal to the Minimum Programming Separation Time (`XcpMinSTPgm`).
Multiplicity	1..1
Type	INTEGER
Configuration class	**PreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpAddressGranularity**
Label	Address Granularity
Description	Defines the size of an element contained at a single address (ASAM parameter: `AG`). Note: This parameter is not fully supported in the current implementation. The current implementation supports only BYTE granularity.
Multiplicity	1..1
Type	ENUMERATION
Default value	BYTE
Range	BYTE WORD DWORD
Configuration class	**VariantPreCompile:** VariantPreCompile

Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpGranularityOdtEntrySizeDaq**	
Label	Granularity for ODT Entry Size - DAQ	
Description	Defines the smallest size of a data element referenced by an ODT entry with DAQ direction (ASAM parameter: `GRANULARITY_ODT_ENTRY_SIZE_DAQ`). Note: This parameter is not fully supported in the current implementation. The current implementation supports only BYTE granularity.	
Multiplicity	1..1	
Type	ENUMERATION	
Default value	BYTE	
Range	BYTE WORD DWORD DLONG	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**XcpGranularityOdtEntrySizeStim**	
Label	Granularity for ODT Entry Size - STIM	
Description	Defines the smallest size of a data element referenced by an ODT entry with STIM direction (ASAM parameter: `GRANULARITY_ODT_ENTRY_SIZE_STIM`). Note: This parameter is not fully supported in the current implementation. The current implementation supports only BYTE granularity.	
Multiplicity	1..1	
Type	ENUMERATION	
Default value	BYTE	
Range	BYTE WORD DWORD DLONG	
Configuration class	**VariantPreCompile:**	VariantPreCompile

Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpUserHeader**	
Label	User Header File	
Description	Defines the header file for user-defined symbols. This file should provide the Xcp with the declarations of variables specified by the ODT Entry Addresses (`XcpOdtEntryAddress`). Note: If the header file for user-defined symbols is not provided, to avoid compilation errors, this parameter should be empty.	
Multiplicity	1..1	
Type	STRING	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**XcpMasterBlockModeSupported**	
Label	Master Block Mode Support (Download)	
Description	Enables the master block mode functionality. Affected command(s): ► DOWNLOAD ► DOWNLOAD_NEXT **Optimization Effect:** ► **RAM reduction (code):** Disabling this parameter reduces the RAM consumption of the module code. ► **ROM reduction (code):** Disabling this parameter reduces the ROM consumption of the module code. ► **Execution time reduction (code):** Disabling this parameter reduces the execution time of the module code.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**XcpMinST**

Label	Minimum Separation Time [100us]
Description	Defines the required minimum separation time between the packets of a block transfer from the master device to the slave device in units of 100 microseconds (ASAM parameter: `MIN_ST`). Range: ► 0 .. 255 Dependency on parameter(s): ► Master Block Mode Support (`XcpMasterBlockModeSupported`): The parameter is used only if Master Block Mode Support is enabled. ► Main Function Period (`XcpMainFunctionPeriod`): This parameter must be lower than or equal to the Minimum Separation Time (`XcpMinST`).
Multiplicity	1..1
Type	INTEGER
Configuration class	**PreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpMaxBS**
Label	Maximum Blocks
Description	**This parameter is only used as the default value of the connection-specific XcpMaxBS.** Defines the maximum number of blocks in a DOWNLOAD block sequence. Includes the initial DOWNLOAD command (ASAM parameter: `MAX_BS`). Range: ► 1 .. 255 **Optimization Effect:** ► **RAM reduction (code):** Selecting a smaller value for this parameter reduces the RAM consumption of the module code.
Multiplicity	1..1
Type	INTEGER
Configuration class	**PreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	XcpTimestampFixed	
Label	Timestamp Fixed	
Description	Defines whether the Xcp slave always sends DTO packets in timestamped mode (ASAM parameter: `TIMESTAMP_FIXED`). Dependency on parameter(s): ► DAQ list support (`XcpDaqSupported`): must be enabled in order for this parameter to be editable. ► Timestamp Type (`XcpTimestampType`): must be different than NO_TIME_-STAMP in order for this parameter to be editable.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**PreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	XcpUserTimestampSupported
Label	User Timestamp Support
Description	Enables the user timestamp feature. ► true: The user-specific callout function `Xcp_ApplGetTimestamp()` is called to get the current timestamp. ► false: The Os counter referred to by parameter `XcpCounterRef` is used to get the current timestamp. Dependency on parameter(s): ► DAQ list support (`XcpDaqSupported`): must be enabled in order for this parameter to be editable. ► Timestamp Type (`XcpTimestampType`): must be different than NO_TIME_-STAMP in order for this parameter to be editable. Note: Refer to the module documentation for the prototype of the callout function `Xcp_ApplGetTimestamp()`. **Optimization Effect:** ► **ROM reduction (config):** Disabling this parameter reduces the ROM consumption of the module configuration.
Multiplicity	1..1

Type	BOOLEAN	
Default value	false	
Configuration class	**PreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**XcpStoreDaqSupported**
Label	Store DAQ Lists Support
Description	Enables the support for storing the DAQ lists in non-volatile (NV) memory. Dependency on parameter(s): ► DAQ list support (`XcpDaqSupported`): must be enabled to support the use of DAQ lists. ► XCP on Ethernet (`XcpOnEthernetEnabled`): must be disabled to support the storage of DAQ lists. ► NvM block for DAQ storage (`XcpStoreDaqNvMBlock`): must reference an NvM block that is configured in the NVRAM manager module (NvM) for DAQ list storage. **Optimization Effect:** ► **ROM reduction (code):** Disabling this parameter reduces the ROM consumption of the module code. ► **ROM reduction (config):** Disabling this parameter reduces the ROM consumption of the module configuration. ► **RAM reduction (config):** Disabling this parameter reduces the RAM consumption of the module configuration. ► **Execution time reduction (code):** Disabling this parameter reduces the execution time of the module code.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**PreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpStoreDaqNvMBlock**
Label	NvM Block for DAQ List Storage

Description	Reference to the NVRAM Manager (NvM) block descriptor that is used to store the DAQ configuration. The NvM block referenced by this parameter must be configured with the following mandatory values: ► `NvMNvBlockLength`: The length must be given as the C code `sizeof()` of the `Xcp_DaqLists` structure. One way to determine the exact size is to check the length of the symbol `Xcp_DaqLists` in the map file generated by the compiler. Note that in some rare cases, e.g. depending on the compiler settings, the value of sizeof(Xcp_DaqLists) is not the same as the value from the map file. The other way to obtain the value is to place a breakpoint in the `Xcp_InitializeDaqLists()` function (Xcp.c) where the corresponding DET error is thrown. ► `NvMBlockWriteProt`: false ► `NvMSelectBlockForReadall`: true ► `NvMSingleBlockCallback`: address to the `Xcp_NvmStoreDaqSingleCbk()` function If RAM optimizations are disabled, the NvMRamBlockDataAddress parameter must be configured as follows: ► `NvMRamBlockDataAddress`: address to the runtime DAQ lists structure (`XCP_NVM_DAQ_LISTS_RAM_ADDRESS`) If RAM optimizations are enabled, the NvMRamBlockDataAddress parameter must not be configured and instead the following parameters must be configured as follows: ► `NvMReadRamBlockFromNvCallback`: Xcp_ReadRamBlockFromNvMNativeCbk ► `NvMWriteRamBlockToNvCallback`: Xcp_WriteRamBlockToNvMNativeCbk The NvM block referenced by this parameter is recommended to be configured with the following values: ► `NvMBlockUseCrc`: true ► `NvMWriteBlockOnce`: false Dependency on parameter(s): ► Store DAQ lists support (`XcpStoreDaqSupported`): must be enabled if DAQ lists are to be stored in the NV memory.

Multiplicity	1..1
Type	REFERENCE
Configuration class	**PreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpTxRetryCount**
Label	Tx Retry Count
Description	Defines the number of times the data is retried for transmission. Range: ► 0 .. 255 **Optimization Effect:** ► **ROM reduction (code):** Setting the value ZERO for this parameter reduces the ROM consumption of the module code. ► **RAM reduction (code):** Setting the value ZERO for this parameter reduces the RAM consumption of the module code.
Multiplicity	1..1
Type	INTEGER
Default value	0
Range	<=255 >=0
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpTxBusTimeout**
Label	Tx Bus Timeout[s]
Description	Defines the maximum allowed time frame for a message to be transmitted from the Xcp slave to the Xcp master. If no retries (`XcpTxBusRetry`) are configured, when the timeout occurs, the Xcp slave autonomously disconnects. Otherwise, the message is re-transmitted for the configured amount of retries. Each re-transmission attempt resets the configured timeout. Range: ► 0.001 .. 255

	Note: In order to have normal communication behavior between the Xcp master and the Xcp slave, it is recommended to configure the same timeout value for both the Xcp master and the Xcp slave.
Multiplicity	1..1
Type	FLOAT
Default value	2.0
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpTxBusRetry**
Label	Tx Bus Retry
Description	Defines the number of retries that the Xcp slave performs in order to transmit a PDU. Whenever a PDU is accepted to be transmitted by the lower layer, the Xcp slave waits for `XcpTxBusTimeout` time to receive the confirmation of successful transmission. If the timeout occurs, the Xcp slave retries the whole lost PDU transmission for the configured number of times. When the number of retries expires (if it is not configured with the reserved value of 255 - infinite retries), the Xcp slave autonomously disconnects from the connection. Range: ► 0 .. 254 ► 255: reserved value for infinite number of retries Note: Do not confuse this parameter with `XcpTxRetryCount`, which retries only the transmit request rejected by the interface layer (i.e. calls to `xxxIf_Transmit()`). **Optimization Effect:** ► **ROM reduction (code):** Disabling this parameter reduces the ROM consumption of the module code. ► **RAM reduction (code):** Disabling this parameter reduces the RAM consumption of the module code.
Multiplicity	1..1
Type	INTEGER
Default value	0

Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**XcpGetIdType1Callout**
Label	User defined callout for GET_ID
Description	Defines whether the user-defined callout for the command GET_ID type 1 is supported. Note: The user-defined callout shall have the following prototype: ► `Xcp_ApplReturnType Xcp_ApplGetIdentification(uint8* IdBufferPtr, uint8* IdLength)` The possible return value is: XCP_APPL_OK Function successful. Any other return value is interpreted as an error in retrieving the identification information.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpGetIdType1MaxLength**
Label	Maximum size for the identification information
Description	Defines the maximum size for user-defined identification information, in bytes. Range: ► 1 .. 0xFF
Multiplicity	1..1
Type	INTEGER
Range	<=255 >=1
Configuration class	**PreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpASAMMC2Filename**
Label	ASAM-MC2 File Name

Description	Defines the ASAM-MC2 filename.	
Multiplicity	0..1	
Type	STRING	
Configuration class	**PreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**XcpSeedAndKeyEnabled**
Label	Seed And Key Support
Description	Enables resource protection, i.e. support for the commands *GET_SEED* and *UNLOCK*. To use resource protection, proceed as follows: 1. Enable resource protection. 2. Implement the two callout functions `Xcp_ApplCompareKey()` and `Xcp_ApplGetSeed()`. Note: Refer to the module documentation for the prototypes of the callout functions `Xcp_ApplCompareKey()` and `Xcp_ApplGetSeed()`. **Optimization Effect:** ► **ROM reduction (code):** Disabling this parameter reduces the ROM consumption of the module code. ► **RAM reduction (code):** Disabling this parameter reduces the RAM consumption of the module code.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpEventPacketEnabled**
Label	Event (EV) Packet Support
Description	Enables the transmission of event (EV) packets from the slave device to the master device. Note: This parameter is not fully supported in the current implementation. The following event packets are implemented

	► *EV_DAQ_OVERLOAD* ► *EV_STIM_TIMEOUT* ► *EV_STORE_DAQ* ► *EV_CLEAR_DAQ* ► *EV_STORE_CAL* ► *EV_RESUME_MODE* ► *EV_CMD_PENDING* Note: ► If `XcpSupportForOverloadMSB` is enabled, the overload notification is performed, towards the master, exclusively by setting the most significant bit from PID for the affected DTO package, even if the `XcpEventPacketEnabled` parameter is enabled. ► In case `XcpBswDistribution` is enabled, EV_RESUME_MODE, EV_CLEAR_DAQ and EV_STORE_DAQ events are not available. **Optimization Effect:** ► **ROM reduction (code):** Disabling this parameter reduces the ROM consumption of the module code. ► **RAM reduction (code):** Disabling this parameter reduces the RAM consumption of the module code. ► **Execution time reduction (code):** Disabling this parameter reduces the execution time of the module code.
Multiplicity	1..1
Type	BOOLEAN
Default value	true
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpModifyBitsSupported**
Label	MODIFY_BITS Command Support
Description	Enables the optional command `MODIFY_BITS`. ► true: The command `MODIFY_BITS` is supported. ► false: The command `MODIFY_BITS` is not supported. **Optimization Effect:**

	► **ROM reduction (code):** Disabling this parameter reduces the ROM consumption of the module code.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpWriteDaqMultipleSupported**
Label	WRITE_DAQ_MULTIPLE Command Support
Description	Enables the optional command `WRITE_DAQ_MULTIPLE`. ► true: The command `WRITE_DAQ_MULTIPLE` is supported. ► false: The command `WRITE_DAQ_MULTIPLE` is not supported. Note that on connections where XcpMaxCto is less than 10, this command is rejected with an ERR_CMD_SYNTAX error. **Optimization Effect:** ► **ROM reduction (code):** Disabling this parameter reduces the ROM consumption of the module code.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpBuildChecksumSupport**
Label	Build Checksum Support
Description	Enables the optional command `BUILD_CHECKSUM`. ► true: The command `BUILD_CHECKSUM` is supported. ► false: The command `BUILD_CHECKSUM` is not supported. **Optimization Effect:** ► **ROM reduction (code):** Disabling this parameter reduces the ROM consumption of the module code.

Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**XcpBuildChecksumType**
Label	Checksum Type
Description	Defines the checksum type to be used for the optional command `BUILD_CHECKSUM`. Range: ► `XCP_CRC_16_CITT`: The AUTOSAR standard CRC 16 CITT algorithm is used. ► `XCP_CRC_32`: The AUTOSAR standard CRC 32 algorithm is used. ► `XCP_CRC_USER_CALLOUT`: The user-specific callout function `Xcp_ApplBuildChecksum()` is called to compute the checksum. Note: Refer to the module documentation for the prototype of the callout function `Xcp_ApplBuildChecksum()`. Dependency on parameter(s): ► Build Checksum Support (`XcpBuildChecksumSupport`) must be enabled. **Optimization Effect:** ► **ROM reduction (code):** Choosing a different value than XCP_CRC_USER_CALLOUT for this parameter reduces the ROM consumption of the module code.
Multiplicity	1..1
Type	ENUMERATION
Default value	XCP_CRC_16_CITT
Range	XCP_CRC_16_CITT XCP_CRC_32 XCP_CRC_USER_CALLOUT
Configuration class	**PreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	XcpBuildChecksumMainFunctionSupport	
Label	Build Checksum on Xcp_CrcMainFunction	
Description	Enables separately Build Checksum Main Function. ► true: Enables support for the asynchronous calculation of the build checksum in the Xcp_CrcMainFunction context. If this parameter is enabled, the Xcp_CrcMainFunction runnable is provided for asynchronous calculation of the build checksum operation. ► false: Disables separately Build Checksum Main Function. Dependency on parameter(s): ► Build Checksum Support (`XcpBuildChecksumSupport`) must be enabled.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**PreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	XcpCRCMainFunctionPeriod	
Label	CRC Main Function Period [s]	
Description	Defines the time interval in which the Xcp_CrcMainFunction is invoked by the BSW Scheduler. Range: ► 0.001 .. 255 Dependency on parameter(s): ► Asynchronous Build Checksum Support (`XcpBuildChecksumMainFunctionSupport`) must be enabled.	
Multiplicity	1..1	
Type	FLOAT	
Configuration class	**PreCompile:**	VariantPreCompile

Parameter Name	XcpAddressTranslationSupport
Label	Address Translation Callout Support
Description	Enables the address translation feature.

	► true: Address Translation is enabled. The user-specific callout function `Xcp_ApplGetAddress()` is called to translate the address. This callout function must be aware of any runtime changes made regarding the slave's memory (e.g.: the active page being changed by a page switching command) in order to perform a correct address translation. ► false: Address Translation is not enabled. The module directly uses the 32-bit address as the physical address for accessing the memory. The address extension is ignored and no address translation is performed. Note: ► The 8-bit address extension is not supported. ► Refer to the module documentation for the prototype of the callout function `Xcp_ApplGetAddress()`. **Optimization Effect:** ► **ROM reduction (code):** Disabling this parameter reduces the ROM consumption of the module code. ► **Execution time reduction (code):** Disabling this parameter reduces the execution time of the module code.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpSetMtaEndianness**
Label	SET_MTA endianness support

Description	Enables support for platform endianness when processing the address DWORD (bytes from position 4 to 7) within a SET_MTA command.
	► true: Byte 4 of the SET_MTA command always represents the least significant byte of the final address.
	► false:
	  ► On big-endian platforms, byte 4 of the SET_MTA command represents the most significant byte of the final address.
	  ► On little-endian platforms, byte 4 of the SET_MTA command represents the least significant byte of the final address.

Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**XcpBlockWriteReadMemoryRAMMechanism**
Label	Enable Write/Read RAM Memory Check
Description	Enables the memory access callout features. ► true: ► Memory access callout feature "Write data in RAM Memory without MemCopy" is enabled. The user-specific callout function specified in "Write to RAM Callback Function" is called to write data to RAM memory. ► Memory access callout feature "Read data from RAM Memory without MemCopy" is enabled. The user-specific callout function specified in "Read from RAM Callback Function" is called to read data from RAM memory. ► false: Memory access callout features "Write data in RAM Memory without MemCopy" and "Read data from RAM Memory without MemCopy" are disabled. Data shall be copied using TS_MemCpy().
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpWriteMemoryRAMCallback**
Label	Write to RAM Callout Function
Description	Specifies the name of a user-defined callout function that is called by the Xcp, during the processing of download, download next, and modify bits commands, for writing data to RAM. The callout function shall have the following prototype: ► `Xcp_ApplReturnType Xcp_ApplWriteDataToRAM(void* AddressPtr, uint8* DataPtr, uint8 DataLength)`

	The possible return values are: ► XCP_APPL_OK: Function successful ► XCP_APPL_ERR_OUT_OF_RANGE: Parameter is invalid Options: ► name of callout function: this function is called to write data to RAM block ► empty: no function is called Note: ► The declaration and definition of this function in user template files is the default name `Xcp_ApplWriteDataToRAM()`. If this name is changed at configuration phase, you must provide also the declaration of this function within a user header file. More explicitly, if you declare this callout function as it is specified in the template user file: Xcp_ApplWriteDataToRAM, then you must include the template user header file Xcp_UserCallouts.h in the configuration parameter XcpUserHeader. If you define the name of the callout function, then a header file containing the declaration of the callout function must be included in the configuration parameter XcpUserHeader.
Multiplicity	1..1
Type	FUNCTION-NAME
Default value	Xcp_ApplWriteDataToRAM
Configuration class	**PreCompile:** VariantPreCompile
Origin	AUTOSAR_ECUC

Parameter Name	**XcpReadMemoryRAMCallback**
Label	Read from RAM Callout Function
Description	Specifies the name of a user-defined callout function that is called by the Xcp during the processing of upload, short upload, and modify bits commands, for reading data from RAM. The callout function must have the following prototype: ► `Xcp_ApplReturnType Xcp_ApplReadDataFromRAM(void* AddressPtr, uint8* DataPtr, uint8 DataLength)` The possible return values are: ► XCP_APPL_OK: Function successful ► XCP_APPL_ERR_OUT_OF_RANGE: Parameter is invalid

	Options: ► name of call-out function: this function is called to read data from a RAM block ► empty: no function is called Note: ► The declaration and definition of this function in user template files is the default name `Xcp_ApplReadDataFromRAM()`. If this name is changed at configuration phase, you must provide also the declaration of this function within a user header file. More explicitly, if you declare this callout function as it is specified in the template user file: Xcp_ApplReadDataFromRAM, then you must include the template user header file Xcp_UserCallouts.h in the configuration parameter XcpUserHeader. If you define the name of the callout function, then a header file containing the declaration of the callout function must be included in the configuration parameter XcpUserHeader.
Multiplicity	1..1
Type	FUNCTION-NAME
Default value	Xcp_ApplReadDataFromRAM
Configuration class	**PreCompile:** VariantPreCompile
Origin	AUTOSAR_ECUC

Parameter Name	**XcpEnableXcpControlApi**
Label	Xcp Enable Xcp Control Api
Description	Enables the control feature. The Xcp_Control() API is available only if this parameter is enabled. ► true: Control functionality is enabled. You can optionally enable or disable the module. ► If you call the control function with the XCP_STATE_ACTIVE command, the module will have full functionality. ► If you call the control function with the XCP_STATE_INACTIVE command, the module will suppress all functionality. ► false: Control is not enabled. The functionality is not available. Note: ► Refer to the module documentation for the prototype of the control function `Xcp_Control()`.

	► *If the module is disabled at precompile time (i.e. both parameters XcpEnableXcpControlApi and XcpDefaultXcpModuleState are set to false), you must not map the main function and events because, with this configuration, the module does not exist.*
Multiplicity	1..1
Type	BOOLEAN
Default value	true
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpDefaultXcpModuleState**
Label	Xcp Default Xcp Module State
Description	Describes the module's default state. ► true: Xcp default state is Enabled. ► The full functionality of the Xcp module is available for usage. ► false: Xcp default state is Disabled. ► The Xcp module exists but the functionality is suppressed. Note: ► Refer to the module documentation for the prototype of the parameter XCP_DEFAULT_XCP_MODULE_STATE. ► *If the module is disabled at precompile time (i.e. both parameters XcpEnableXcpControlApi and XcpDefaultXcpModuleState are set to false), you must not map the main function and events because, with this configuration, the module does not exist.*
Multiplicity	1..1
Type	BOOLEAN
Default value	true
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpSupportForOverloadMSB**
Label	Support for Overload MSB
Description	Enables the support for an overload situation, in which the slave may set the Most Significant Bit (MSB) of the PID of the next successfully transmitted packet.

	When the MSB of the PID is used, the maximum number of (absolute or relative) ODT numbers is limited and must be in the range: 0x00 < ODT_NUMBER(DAQ with overrun_msb) > 0x7C Dependency on parameter(s): ► `XcpDaqSupported` must be enabled in order to have the Xcp Support for overload MSB configurable. ► If `XcpSupportForOverloadMSB` is enabled, the overload notification is performed, towards the master, exclusively by setting the MSB from the PID, for the affected DTO package, even if the `XcpEventPacketEnabled` parameter is enabled. Values for `XcpSupportForOverloadMSB` parameter: ► true: Xcp support for sending the overload notification to the master via the PID's MSB. ► false: The overload notification is not sent to the master via the PID's MSB.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**PreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpSupportForGetDaqId**
Label	Support for CAN or CAN-FD Transport layer GET_DAQ_ID

Description	Enables the support for the GET_DAQ_ID CAN transport layer command.
	
	Dependency on parameter(s):
	
	► `XcpOnCanEnabled` or `XcpOnCanFDEnabled` and `XcpDaqSupported` must be enabled in order to have the support for the transport layer GET_-DAQ_ID command.
	► Values for `XcpSupportForGetDaqId`:
	  ► Enabled: The maximum number of Xcp PDUs, configured for each Xcp connection over CAN or CAN-FD (refer to `XcpConnectionCfg`), supporting DAQ/STIM packages, is restricted to 1, for each direction.
	  ► Disabled: In this case, there is no restriction for the maximum number of Xcp PDUs, configured for each Xcp connection over CAN or CAN-FD (refer to `XcpConnectionCfg`), supporting DAQ/STIM packages.

	Note: All DTOs can be sent/received on each of the configured PDU supporting DAQ/STIM packages.	
Multiplicity	1..1	
Type	BOOLEAN	
Configuration class	**PreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**XcpSupportForFlxTPCommands**
Label	Support for FlexRay Transport Layer Commands
Description	Enables the support for the FlexRay transport layer commands: ► FLX_ASSIGN (Assign/deassign FlexRay LPDU identifiers to buffers) ► FLX_ACTIVATE (Activate communication of a FlexRay buffer) ► FLX_DEACTIVATE (Deactivate communication of a FlexRay buffer) ► GET_DAQ_FLX_BUF (Get DAQ list FlexRay buffer(s)) ► SET_DAQ_FLX_BUF (Set DAQ list FlexRay buffer(s)) Dependency on parameter(s): ► `XcpOnFlexRayEnabled` must be enabled in order to have the support for FlexRay transport layer commands configurable. ► Values for `XcpSupportForFlxTPCommands`: ► Enabled: The Xcp master can use the FlexRay transport layer commands. ► Disabled: The Xcp master cannot use the FlexRay transport layer commands.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**PreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpSupportDynamicComChannels**
Label	Support dynamic communication channels
Description	Enables support for dynamic communication channels. This feature allows you to enable or disable a communication channel at runtime. As this behavior is

	not specified by ASAM, this feature is implemented by reserving a user-defined command, sub-command code 0x00U. Format of the user command to be received: Command_Code SubCommand_Code Direction PduId PduState ► Command Code: 0xF1 - user command ► SubCommand Code: 0x00 - reserved sub-command code ► PDU Direction: 0x00 - transmission; 0x01 - reception ► PduId: Configured index for Xcp PDU ► State: State requested for a specific PDU: 0x00 - Disabled, 0x01 - Enabled Dependency on parameter(s): ► `XcpUserCommand` sub-command code 0x00 must be configured as below: ► `XcpSubCommandCode`: 0 ► `XcpSubCommandLength`: 3 ► `XcpSubCommandResponseLength`: 0 ► `XcpSubCommandCallout`: Xcp_EnableCommunicationChannel ► Values for `XcpSupportDynamicComChannels`: ► Enabled: You can use a reserved user command in order to enable or disable Xcp PDUs at runtime. ► Disabled: All configured Xcp PDUs are static.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpSupportComMainFunctions**
Label	Support for Tx and Rx MainFunctions
Description	Defines whether additional main processing functions shall be used for transmission and reception in Xcp. If this feature is enabled, `Xcp_MainFunctionTx` and `Xcp_MainFunctionRx` are scheduled functions with the periodicity defined by this parameter. If this parameter is enabled, the value configured for the periodicity has to be higher than 0.0.

Multiplicity	0..1	
Type	FLOAT	
Configuration class	**PreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**XcpSupportForPDURoutingTable**	
Label	Support for PDU routing	
Description	Enables the support for an Xcp-internal PDU routing table. If this parameter is enabled, a table for switching from lower layer PduId to Xcp PduId is generated. This option should be used if callback functions from the lower layer get as parameter the PduId of the lower layer and not the PduId of Xcp. ► Values for `XcpSupportForPDURoutingTable`: ► Enabled: Callback functions from the lower layer are called with the PduId of the lower layer. A table for switching from lower layer PduId to Xcp PduId is generated in Xcp. ► Disabled: Callback functions from the lower layer are called with the Xcp PduId. A table for switching from lower layer PduId to Xcp PduId is not generated in Xcp.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**XcpServiceWrapper**	
Label	XCP Service Wrapper	
Description	By enabling this feature, the Xcp module generates a wrapper. The wrapper functions must be called by all other modules, and all Xcp-external interfaces must only be called by the wrapper. Xcp only exports the wrapper schedulable entity and this is the MainFunction that must be mapped in the RTE. ► Values for `XcpServiceWrapper`: ► Enabled: XcpW is generated and is the only caller of Xcp interfaces. The user-defined callout function App_Xcp_Wrapper_IsXcpUsed must be provided within the user header file. ► Disabled: No wrapper is generated - normal use case.	

Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**XcpFlexRaySyncTimeout**	
Label	FlexRay Sync Timeout[s]	
Description	Defines the maximum allowed time frame until the Xcp slave disconnects from the master if a loss of FlexRay synchronization occurs. FlexRay synchronization is retried at each main function cycle execution. This option is used to prevent a freeze situation that can occur when FlexRay synchronization continues to fail indefinitely.	
Multiplicity	0..1	
Type	FLOAT	
Default value	2.0	
Configuration class	**PreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**XcpMulticastCommands**	
Label	Multicast Commands Support	
Description	Enables multicast commands. The only supported multicast command is GET_SLAVE_ID over Ethernet UDP. Note: At least one Ethernet Ipv4 UDP connection needs to be configured in order to use this option.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**XcpMulticastToUnicastRetry**
Label	Tx Multicast to Unicast Retry

Description	Defines the number of retries that the Xcp slave performs to change back the address and port to unicast, before it autonomously disconnects from the master. The change to unicast is retried at each main function cycle execution.	
Multiplicity	1..1	
Type	INTEGER	
Default value	0	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**XcpSynchReceivedCallout**	
Label	Synch Received Callout	
Description	Enables the usage of the user callout function Xcp_ApplSynchReceived. Xcp calls this function when a SYNCH command is received.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPreCompile:**	VariantPreCompile

5.2.1.34. XcpUserCommand

Parameters included	
Parameter name	**Multiplicity**
XcpSubCommandCode	1..1
XcpSubCommandLength	1..1
XcpSubCommandResponseLength	1..1
XcpSubCommandCallout	1..1
XcpSubCommandSeedSupport	1..1

Parameter Name	**XcpSubCommandCode**
Description	This parameter indicates sub-command code.
Multiplicity	1..1
Type	INTEGER

Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**XcpSubCommandLength**	
Description	Represents the number of bytes to be sent by the master within the command. It does not include the number of bytes reserved for the command and sub-command code. It consists of the number of bytes that you want to send as parameters for a specific sub-command.	
Multiplicity	1..1	
Type	INTEGER	
Default value	0	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**XcpSubCommandResponseLength**	
Description	Represents the number of bytes to be received by the master within the command. It does not include the number of bytes reserved for a positive response. It only consists of the number of data bytes to be sent from slave to the master as response.	
Multiplicity	1..1	
Type	INTEGER	
Default value	0	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**XcpSubCommandCallout**
Label	User Defined Sub-Command Callout
Description	Specifies the name of a user-defined callout function to be called by Xcp for a specific user command. The callout function must have the following prototype: ► `FUNC (uint8, XCP_CODE) XXX_UserCallout( P2CONST( uint8, AUTOMATIC, XCP_APPL_DATA ) ParamPtr, uint8 Length, P2VAR( uint8, AUTOMATIC, XCP_APPL_DATA ) SubCmdResponse, uint8 RespLength )` The possible return value is: E_OK Function successful

	For any other value, the command is ignored.	
Multiplicity	1..1	
Type	FUNCTION-NAME	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**XcpSubCommandSeedSupport**	
Description	Indicates the type of resource to be unlocked for executing the sub-command.	
Multiplicity	1..1	
Type	ENUMERATION	
Default value	RESOURCE_UNLOCKED	
Range	RESOURCE_PGM	
	RESOURCE_STIM	
	RESOURCE_DAQ	
	RESOURCE_CAL_PAG	
	RESOURCE_UNLOCKED	
Configuration class	**PreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

5.2.2. Recommended configurations

5.2.2.1. XcpRecConfigurationMaxAllConnections

Containers included	
Container name	**Container definition**
CommonPublishedInformation	CommonPublishedInformation
XcpConfig_0	XcpConfig
XcpGeneral	XcpGeneral
ReportToDem	ReportToDem
XcpDefensiveProgramming	XcpDefensiveProgramming

Parameters included	
Parameter name	**Value**
IMPLEMENTATION_CONFIG_VARIANT	VariantPreCompile

5.2.2.1.1. CommonPublishedInformation

Parameters included	
Parameter name	**Value**
ArMajorVersion	2
ArMinorVersion	0
ArPatchVersion	0
SwMajorVersion	2
SwMinorVersion	12
SwPatchVersion	2
ModuleId	212
VendorId	1
Release	$3SOFT_DELIVERY_VERSION_INFORMATION$

5.2.2.1.2. XcpConfig_0

Containers included	
Container name	**Container definition**
XcpDemEventParameterRefs	XcpDemEventParameterRefs
XcpEventChannel_0	XcpEventChannel
XcpEventChannel_1	XcpEventChannel
XcpEventChannel_2	XcpEventChannel
XcpPdu_Rx_CAN_1	XcpPdu
XcpPdu_Rx_CAN_2	XcpPdu
XcpPdu_Rx_FLexRay_1	XcpPdu
XcpPdu_Tx_CAN_1	XcpPdu
XcpPdu_Tx_CAN_2	XcpPdu
XcpPdu_Tx_FlexRay_1	XcpPdu
XcpPdu_Tx_FlexRay_2	XcpPdu

Containers included	
XcpPdu_Tx_Ethernet_1	XcpPdu
XcpPdu_Rx_Ethernet_1	XcpPdu
XcpConnectionCfg_0	XcpConnectionCfg
XcpConnectionCfg_1	XcpConnectionCfg
XcpConnectionCfg_2	XcpConnectionCfg
XcpConnectionCfg_3	XcpConnectionCfg
XcpMemoryAccessArea	XcpMemoryAccessArea

Parameters included	
Parameter name	**Value**

5.2.2.1.3. XcpDemEventParameterRefs

Parameters included	
Parameter name	**Value**

5.2.2.1.4. XcpEventChannel_0

Parameters included	
Parameter name	**Value**
XcpEventChannelConsistency	DAQ
XcpEventChannelType	DAQ_STIM
XcpEventChannelMaxDaqList	1
XcpEventChannelNumber	0
XcpEventChannelPriority	10
XcpEventChannelTimeCycle	255
XcpEventChannelTimeUnit	TIMESTAMP_UNIT_1S

5.2.2.1.5. XcpEventChannel_1

Parameters included	
Parameter name	**Value**
XcpEventChannelConsistency	EVENT

Parameters included	
XcpEventChannelMaxDaqList	5
XcpEventChannelNumber	1
XcpEventChannelPriority	0
XcpEventChannelTimeCycle	0
XcpEventChannelTimeUnit	TIMESTAMP_UNIT_1MS
XcpEventChannelType	DAQ

5.2.2.1.6. XcpEventChannel_2

Parameters included	
Parameter name	**Value**
XcpEventChannelConsistency	DAQ
XcpEventChannelMaxDaqList	1
XcpEventChannelNumber	2
XcpEventChannelPriority	1
XcpEventChannelTimeCycle	0
XcpEventChannelTimeUnit	TIMESTAMP_UNIT_1MS
XcpEventChannelType	DAQ

5.2.2.1.7. XcpPdu_Rx_CAN_1

Containers included	
Container name	**Container definition**
XcpRxPdu	XcpRxPdu
XcpTxPdu	XcpTxPdu

Parameters included	
Parameter name	**Value**

5.2.2.1.8. XcpRxPdu

Parameters included	
Parameter name	**Value**

Parameters included	
XcpRxPduId	0

5.2.2.1.9. XcpTxPdu

Parameters included	
Parameter name	**Value**
XcpTxPduId	0

5.2.2.1.10. XcpPdu_Rx_CAN_2

Containers included	
Container name	**Container definition**
XcpRxPdu	XcpRxPdu
XcpTxPdu	XcpTxPdu

Parameters included	
Parameter name	**Value**

5.2.2.1.11. XcpRxPdu

Parameters included	
Parameter name	**Value**
XcpRxPduId	1

5.2.2.1.12. XcpTxPdu

Parameters included	
Parameter name	**Value**
XcpTxPduId	0

5.2.2.1.13. XcpPdu_Rx_FLexRay_1

Containers included	
Container name	**Container definition**

Containers included	
XcpRxPdu	XcpRxPdu
XcpTxPdu	XcpTxPdu

Parameters included	
Parameter name	**Value**

5.2.2.1.14. XcpRxPdu

Parameters included	
Parameter name	**Value**
XcpRxPduId	2

5.2.2.1.15. XcpTxPdu

Parameters included	
Parameter name	**Value**
XcpTxPduId	1

5.2.2.1.16. XcpPdu_Tx_CAN_1

Containers included	
Container name	**Container definition**
XcpRxPdu	XcpRxPdu
XcpTxPdu	XcpTxPdu

Parameters included	
Parameter name	**Value**

5.2.2.1.17. XcpRxPdu

Parameters included	
Parameter name	**Value**
XcpRxPduId	1

5.2.2.1.18. XcpTxPdu

Parameters included	
Parameter name	**Value**
XcpTxPduId	0

5.2.2.1.19. XcpPdu_Tx_CAN_2

Containers included	
Container name	**Container definition**
XcpRxPdu	XcpRxPdu
XcpTxPdu	XcpTxPdu

Parameters included	
Parameter name	**Value**

5.2.2.1.20. XcpRxPdu

Parameters included	
Parameter name	**Value**
XcpRxPduId	3

5.2.2.1.21. XcpTxPdu

Parameters included	
Parameter name	**Value**
XcpTxPduId	1

5.2.2.1.22. XcpPdu_Tx_FlexRay_1

Containers included	
Container name	**Container definition**
XcpRxPdu	XcpRxPdu

Containers included	
XcpTxPdu	XcpTxPdu

Parameters included	
Parameter name	**Value**

5.2.2.1.23. XcpRxPdu

Parameters included	
Parameter name	**Value**
XcpRxPduId	5

5.2.2.1.24. XcpTxPdu

Parameters included	
Parameter name	**Value**
XcpTxPduId	2

5.2.2.1.25. XcpPdu_Tx_FlexRay_2

Containers included	
Container name	**Container definition**
XcpRxPdu	XcpRxPdu
XcpTxPdu	XcpTxPdu

Parameters included	
Parameter name	**Value**

5.2.2.1.26. XcpRxPdu

Parameters included	
Parameter name	**Value**
XcpRxPduId	5

5.2.2.1.27. XcpTxPdu

Parameters included	
Parameter name	**Value**
XcpTxPduId	3

5.2.2.1.28. XcpPdu_Tx_Ethernet_1

Containers included	
Container name	**Container definition**
XcpRxPdu	XcpRxPdu
XcpTxPdu	XcpTxPdu

Parameters included	
Parameter name	**Value**

5.2.2.1.29. XcpRxPdu

Parameters included	
Parameter name	**Value**
XcpRxPduId	4

5.2.2.1.30. XcpTxPdu

Parameters included	
Parameter name	**Value**
XcpTxPduId	4

5.2.2.1.31. XcpPdu_Rx_Ethernet_1

Containers included	
Container name	**Container definition**
XcpRxPdu	XcpRxPdu
XcpTxPdu	XcpTxPdu

Parameters included	
Parameter name	**Value**

5.2.2.1.32. XcpRxPdu

Parameters included	
Parameter name	**Value**
XcpRxPduId	3

5.2.2.1.33. XcpTxPdu

Parameters included	
Parameter name	**Value**
XcpTxPduId	4

5.2.2.1.34. XcpConnectionCfg_0

Containers included	
Container name	**Container definition**
XcpTxPduConnectionInfo_0	XcpTxPduConnectionInfo
XcpRxPduConnectionInfo_0	XcpRxPduConnectionInfo

Parameters included	
Parameter name	**Value**
XcpConnectionId	0

5.2.2.1.35. XcpConnectionInterfaceType

Containers included	
Container name	**Container definition**
XcpConnectionOverCAN	XcpConnectionInterfaceType XcpConnectionOverCAN
XcpConnectionOverCANFD	XcpConnectionInterfaceType XcpConnectionOver-CANFD
XcpConnectionOverFlexRay	XcpConnectionInterfaceType XcpConnectionOver-FlexRay

Containers included	
XcpConnectionOverEthernet	XcpConnectionInterfaceType XcpConnectionOverEthernet

Parameters included	
Parameter name	**Value**

5.2.2.1.36. XcpConnectionOverCAN

Parameters included	
Parameter name	**Value**

5.2.2.1.37. XcpConnectionOverCANFD

Parameters included	
Parameter name	**Value**
XcpCanFdMaxDlcRequired	
XcpCanFdMaxDlc	64
XcpCanFdFillValue	255

5.2.2.1.38. XcpConnectionOverFlexRay

Parameters included	
Parameter name	**Value**
XcpFlxNodeAddress	0
XcpFlxHeaderAlignment	PACKET_ALIGNMENT_8
XcpPackMultiMsgInOneFlexRayFrame	false
XcpMaxFlexMsgLengthInit	254

5.2.2.1.39. XcpConnectionOverEthernet

Parameters included	
Parameter name	**Value**
XcpAutoOpenSoCon	true

Parameters included	
XcpPackMultiMsgInOneEthernetFrame	false
XcpMaxEthernetMsgLengthInit	48

5.2.2.1.40. XcpTxPduConnectionInfo_0

Parameters included	
Parameter name	**Value**
XcpPduSupportForResErrCtoType	true
XcpPduSupportForEvServCtoType	true
XcpPduSupportForDaqDtoType	true
XcpPduSupportTxFromTxConfirmation	true
XcpDefaultStateDynamicTxPDU	**(DISABLED)**

5.2.2.1.41. XcpRxPduConnectionInfo_0

Parameters included	
Parameter name	**Value**
XcpRxLowerLayerHandleId	0
XcpPduSupportForCmdCtoType	true
XcpPduSupportForStimDtoType	true
XcpDefaultStateDynamicRxPDU	**(DISABLED)**
XcpPduSupportRxFromRxIndication	true

5.2.2.1.42. XcpConnectionCfg_1

Containers included	
Container name	**Container definition**
XcpTxPduConnectionInfo_0	XcpTxPduConnectionInfo
XcpTxPduConnectionInfo_1	XcpTxPduConnectionInfo
XcpRxPduConnectionInfo_0	XcpRxPduConnectionInfo

Parameters included	
Parameter name	**Value**

Parameters included	
XcpConnectionId	1

5.2.2.1.43. XcpConnectionInterfaceType

Containers included	
Container name	**Container definition**
XcpConnectionOverCAN	XcpConnectionInterfaceType XcpConnectionOverCAN
XcpConnectionOverCANFD	XcpConnectionInterfaceType XcpConnectionOver-CANFD
XcpConnectionOverFlexRay	XcpConnectionInterfaceType XcpConnectionOver-FlexRay
XcpConnectionOverEthernet	XcpConnectionInterfaceType XcpConnectionOverEther-net

Parameters included	
Parameter name	**Value**

5.2.2.1.44. XcpConnectionOverCAN

Parameters included	
Parameter name	**Value**

5.2.2.1.45. XcpConnectionOverCANFD

Parameters included	
Parameter name	**Value**
XcpCanFdMaxDlcRequired	
XcpCanFdMaxDlc	64
XcpCanFdFillValue	255

5.2.2.1.46. XcpConnectionOverFlexRay

Parameters included	
Parameter name	**Value**

Parameters included	
XcpFlxNodeAddress	0
XcpFlxHeaderAlignment	PACKET_ALIGNMENT_8
XcpPackMultiMsgInOneFlexRayFrame	true
XcpMaxFlexMsgLengthInit	254

5.2.2.1.47. XcpConnectionOverEthernet

Parameters included	
Parameter name	**Value**
XcpPackMultiMsgInOneEthernetFrame	false
XcpMaxEthernetMsgLengthInit	48
XcpAutoOpenSoCon	true

5.2.2.1.48. XcpTxPduConnectionInfo_0

Parameters included	
Parameter name	**Value**
XcpPduSupportForResErrCtoType	true
XcpPduSupportForEvServCtoType	true
XcpPduSupportForDaqDtoType	true
XcpPduSupportTxFromTxConfirmation	true
XcpDefaultStateDynamicTxPDU	**(DISABLED)**

5.2.2.1.49. XcpTxPduConnectionInfo_1

Parameters included	
Parameter name	**Value**
XcpPduSupportForResErrCtoType	false
XcpPduSupportForEvServCtoType	true
XcpPduSupportForDaqDtoType	true
XcpPduSupportTxFromTxConfirmation	true
XcpDefaultStateDynamicTxPDU	**(DISABLED)**

5.2.2.1.50. XcpRxPduConnectionInfo_0

Parameters included	
Parameter name	**Value**
XcpRxLowerLayerHandleId	0
XcpPduSupportForCmdCtoType	true
XcpPduSupportForStimDtoType	true
XcpDefaultStateDynamicRxPDU	**(DISABLED)**
XcpPduSupportRxFromRxIndication	true

5.2.2.1.51. XcpConnectionCfg_2

Containers included	
Container name	**Container definition**
XcpTxPduConnectionInfo_0	XcpTxPduConnectionInfo
XcpRxPduConnectionInfo_0	XcpRxPduConnectionInfo

Parameters included	
Parameter name	**Value**
XcpConnectionId	2

5.2.2.1.52. XcpConnectionInterfaceType

Containers included	
Container name	**Container definition**
XcpConnectionOverCAN	XcpConnectionInterfaceType XcpConnectionOverCAN
XcpConnectionOverCANFD	XcpConnectionInterfaceType XcpConnectionOver-CANFD
XcpConnectionOverFlexRay	XcpConnectionInterfaceType XcpConnectionOver-FlexRay
XcpConnectionOverEthernet	XcpConnectionInterfaceType XcpConnectionOverEther-net

Parameters included	
Parameter name	**Value**

5.2.2.1.53. XcpConnectionOverCAN

Parameters included	
Parameter name	**Value**

5.2.2.1.54. XcpConnectionOverCANFD

Parameters included	
Parameter name	**Value**
XcpCanFdMaxDlcRequired	
XcpCanFdMaxDlc	64
XcpCanFdFillValue	255

5.2.2.1.55. XcpConnectionOverFlexRay

Parameters included	
Parameter name	**Value**
XcpFlxNodeAddress	0
XcpFlxHeaderAlignment	PACKET_ALIGNMENT_8
XcpPackMultiMsgInOneFlexRayFrame	false
XcpMaxFlexMsgLengthInit	254

5.2.2.1.56. XcpConnectionOverEthernet

Parameters included	
Parameter name	**Value**
XcpAutoOpenSoCon	true
XcpPackMultiMsgInOneEthernetFrame	true
XcpMaxEthernetMsgLengthInit	72

5.2.2.1.57. XcpTxPduConnectionInfo_0

Parameters included	
Parameter name	**Value**

Parameters included	
XcpPduSupportForResErrCtoType	true
XcpPduSupportForEvServCtoType	true
XcpPduSupportForDaqDtoType	true
XcpPduSupportTxFromTxConfirmation	true
XcpDefaultStateDynamicTxPDU	**(DISABLED)**

5.2.2.1.58. XcpRxPduConnectionInfo_0

Parameters included	
Parameter name	**Value**
XcpRxLowerLayerHandleId	0
XcpPduSupportForCmdCtoType	true
XcpPduSupportForStimDtoType	true
XcpDefaultStateDynamicRxPDU	**(DISABLED)**
XcpPduSupportRxFromRxIndication	true

5.2.2.1.59. XcpConnectionCfg_3

Containers included	
Container name	**Container definition**
XcpTxPduConnectionInfo_0	XcpTxPduConnectionInfo
XcpRxPduConnectionInfo_0	XcpRxPduConnectionInfo

Parameters included	
Parameter name	**Value**
XcpConnectionId	3

5.2.2.1.60. XcpConnectionInterfaceType

Containers included	
Container name	**Container definition**

Containers included	
XcpConnectionOverCAN	XcpConnectionInterfaceType XcpConnectionOverCAN
XcpConnectionOverCANFD	XcpConnectionInterfaceType XcpConnectionOver-CANFD
XcpConnectionOverFlexRay	XcpConnectionInterfaceType XcpConnectionOver-FlexRay
XcpConnectionOverEthernet	XcpConnectionInterfaceType XcpConnectionOverEther-net

Parameters included	
Parameter name	**Value**

5.2.2.1.61. XcpConnectionOverCAN

Parameters included	
Parameter name	**Value**

5.2.2.1.62. XcpConnectionOverCANFD

Parameters included	
Parameter name	**Value**
XcpCanFdMaxDlcRequired	true
XcpCanFdMaxDlc	64
XcpCanFdFillValue	255

5.2.2.1.63. XcpConnectionOverFlexRay

Parameters included	
Parameter name	**Value**
XcpFlxNodeAddress	0
XcpFlxHeaderAlignment	PACKET_ALIGNMENT_8
XcpPackMultiMsgInOneFlexRayFrame	false
XcpMaxFlexMsgLengthInit	254

5.2.2.1.64. XcpConnectionOverEthernet

Parameters included	
Parameter name	**Value**
XcpAutoOpenSoCon	true
XcpPackMultiMsgInOneEthernetFrame	false
XcpMaxEthernetMsgLengthInit	48

5.2.2.1.65. XcpTxPduConnectionInfo_0

Parameters included	
Parameter name	**Value**
XcpPduSupportForResErrCtoType	true
XcpPduSupportForEvServCtoType	true
XcpPduSupportForDaqDtoType	true
XcpPduSupportTxFromTxConfirmation	true
XcpDefaultStateDynamicTxPDU	**(DISABLED)**

5.2.2.1.66. XcpRxPduConnectionInfo_0

Parameters included	
Parameter name	**Value**
XcpRxLowerLayerHandleId	1
XcpPduSupportForCmdCtoType	true
XcpPduSupportForStimDtoType	true
XcpDefaultStateDynamicRxPDU	**(DISABLED)**
XcpPduSupportRxFromRxIndication	true

5.2.2.1.67. XcpMemoryAccessArea

Parameters included	
Parameter name	**Value**
XcpMemoryAccessAreasSupport	false

5.2.2.1.68. XcpGeneral

Parameters included	
Parameter name	**Value**
XcpOnCanFDEnabled	true
XcpCalPagSupported	true
XcpDaqSupported	true
XcpStimSupported	true
XcpSlaveBlockModeSupported	true
XcpAddressGranularity	BYTE
XcpGranularityOdtEntrySizeDaq	BYTE
XcpGranularityOdtEntrySizeStim	BYTE
XcpMasterBlockModeSupported	true
XcpMinST	255
XcpTimestampType	FOUR_BYTE
XcpDaqConfigType	DAQ_DYNAMIC
XcpOdtEntriesCount	3
XcpDaqCount	10
XcpPrescalerSupported	true
XcpOdtEntrySizeDaq	2
XcpOdtEntrySizeStim	2
XcpOdtCount	5
XcpIdentificationFieldType	RELATIVE_BYTE
XcpTxRetryCount	255
XcpTimestampFixed	true
XcpTxBusRetry	0
XcpGetIdType1Callout	**(DISABLED)**
XcpGetIdType1MaxLength	**(DISABLED)**
XcpASAMMC2Filename	**(DISABLED)**
XcpDevErrorDetect	true
XcpMainFunctionPeriod	0.0010
XcpMaxCto	8
XcpMaxDto	8

Parameters included	
XcpMaxEventChannel	1
XcpMinDaq	0
XcpOnCanEnabled	true
XcpOnCddEnabled	false
XcpOnEthernetEnabled	true
XcpOnFlexRayEnabled	true
XcpVersionInfoApi	true
XcpUserHeader	
XcpSeedAndKeyEnabled	true
XcpEventPacketEnabled	true
XcpTimestampUnit	TIMESTAMP_UNIT_1MS
XcpTimestampTicks	1000
XcpUserTimestampSupported	true
XcpStoreDaqSupported	true
XcpWriteDaqMultipleSupported	false
XcpBuildChecksumSupport	true
XcpBuildChecksumType	XCP_CRC_16_CITT
XcpCalPagStoreSupported	true
XcpBuildChecksumMainFunctionSupport	false
XcpCRCMainFunctionPeriod	
XcpAddressTranslationSupport	true
XcpSuppressTxSupport	false
XcpMaxBS	43
XcpDynamicAreaSize	2048
XcpModifyBitsSupported	true
XcpMaxCtoPgm	8
XcpEventTriggerMainFunc	true
XcpProcessMultiEventMainFunc	true
XcpDynamicMirrorAreaSize	1024
XcpPgmSupported	false
XcpMasterBlockModePgmSupported	false

Parameters included	
XcpMaxBSPgm	1
XcpMinSTPgm	0
XcpTxBusTimeout	2.0
XcpBlockWriteReadMemoryRAMMechanism	false
XcpWriteMemoryRAMCallback	
XcpReadMemoryRAMCallback	
XcpEnableXcpControlApi	true
XcpDefaultXcpModuleState	true
XcpSupportForOverloadMSB	false
XcpSupportForGetDaqId	false
XcpSupportForFlxTPCommands	false
XcpSupportDynamicComChannels	false
XcpSupportComMainFunctions	**(DISABLED)**
XcpSupportForPDURoutingTable	false
XcpServiceWrapper	false

5.2.2.1.69. ReportToDem

Parameters included	
Parameter name	**Value**
XcpRetryFailedReportToDem	DET
XcpRetryFailedDemDetErrorId	30
XcpIllegalMemoryAccessReportToDem	DISABLE
XcpIllegalMemoryAccessDemDetErrorId	30
XcpPduBufferLockedReportToDem	DISABLE
XcpPduBufferLockedDemDetErrorId	130
XcpRespCTOQueueFullReportToDem	DISABLE
XcpRespCTOQueueFullDemDetErrorId	133
XcpPDUBufferFullReportToDem	DISABLE
XcpPDUBufferFullDemDetErrorId	129
XcpPDULostReportToDem	DISABLE
XcpPDULostDemDetErrorId	131

Parameters included	
XcpStimulationDataLostReportToDem	DISABLE
XcpStimulationDataLostDemDetErrorId	135

5.2.2.1.70. XcpDefensiveProgramming

Parameters included	
Parameter name	**Value**
XcpDefProgEnabled	false
XcpPrecondAssertEnabled	false
XcpPostcondAssertEnabled	false
XcpStaticAssertEnabled	false
XcpUnreachAssertEnabled	false
XcpInvariantAssertEnabled	false

5.2.2.2. XcpRecConfigurationMaxFlexRay

Containers included	
Container name	**Container definition**
CommonPublishedInformation	CommonPublishedInformation
PublishedInformation	PublishedInformation
XcpConfig_0	XcpConfig
XcpGeneral	XcpGeneral
ReportToDem	ReportToDem
XcpDefensiveProgramming	XcpDefensiveProgramming

Parameters included	
Parameter name	**Value**
IMPLEMENTATION_CONFIG_VARIANT	VariantPreCompile

5.2.2.2.1. CommonPublishedInformation

Parameters included	
Parameter name	**Value**

Parameters included	
ArMajorVersion	2
ArMinorVersion	0
ArPatchVersion	0
SwMajorVersion	2
SwMinorVersion	12
SwPatchVersion	2
ModuleId	212
VendorId	1
Release	

5.2.2.2.2. PublishedInformation

Parameters included	
Parameter name	**Value**
PbcfgMSupport	false

5.2.2.2.3. XcpConfig_0

Containers included	
Container name	**Container definition**
XcpDemEventParameterRefs	XcpDemEventParameterRefs
XcpEventChannel_0	XcpEventChannel
XcpPdu_Rx	XcpPdu
XcpPdu_Tx	XcpPdu
XcpConnectionCfg_0	XcpConnectionCfg
XcpMemoryAccessArea	XcpMemoryAccessArea

Parameters included	
Parameter name	**Value**

5.2.2.2.4. XcpDemEventParameterRefs

Parameters included	
Parameter name	**Value**

5.2.2.2.5. XcpEventChannel_0

Parameters included	
Parameter name	**Value**
XcpEventChannelConsistency	ODT
XcpEventChannelMaxDaqList	1
XcpEventChannelNumber	0
XcpEventChannelPriority	0
XcpEventChannelTimeCycle	1
XcpEventChannelTimeUnit	TIMESTAMP_UNIT_1S
XcpEventChannelType	DAQ_STIM

5.2.2.2.6. XcpPdu_Rx

Containers included	
Container name	**Container definition**
XcpRxPdu	XcpRxPdu
XcpTxPdu	XcpTxPdu

Parameters included	
Parameter name	**Value**

5.2.2.2.7. XcpRxPdu

Parameters included	
Parameter name	**Value**
XcpRxPduId	0

5.2.2.2.8. XcpTxPdu

Parameters included	
Parameter name	**Value**
XcpTxPduId	0

5.2.2.2.9. XcpPdu_Tx

Containers included	
Container name	**Container definition**
XcpRxPdu	XcpRxPdu
XcpTxPdu	XcpTxPdu

Parameters included	
Parameter name	**Value**

5.2.2.2.10. XcpRxPdu

Parameters included	
Parameter name	**Value**
XcpRxPduId	1

5.2.2.2.11. XcpTxPdu

Parameters included	
Parameter name	**Value**
XcpTxPduId	0

5.2.2.2.12. XcpConnectionCfg_0

Containers included	
Container name	**Container definition**
XcpTxPduConnectionInfo_0	XcpTxPduConnectionInfo

Containers included	
XcpRxPduConnectionInfo_0	XcpRxPduConnectionInfo

Parameters included	
Parameter name	**Value**
XcpConnectionId	0

5.2.2.2.13. XcpConnectionInterfaceType

Containers included	
Container name	**Container definition**
XcpConnectionOverCAN	XcpConnectionInterfaceType XcpConnectionOverCAN
XcpConnectionOverCANFD	XcpConnectionInterfaceType XcpConnectionOver-CANFD
XcpConnectionOverFlexRay	XcpConnectionInterfaceType XcpConnectionOver-FlexRay
XcpConnectionOverEthernet	XcpConnectionInterfaceType XcpConnectionOverEther-net

Parameters included	
Parameter name	**Value**

5.2.2.2.14. XcpConnectionOverCAN

Parameters included	
Parameter name	**Value**

5.2.2.2.15. XcpConnectionOverCANFD

Parameters included	
Parameter name	**Value**
XcpCanFdMaxDlcRequired	
XcpCanFdMaxDlc	64
XcpCanFdFillValue	255

5.2.2.2.16. XcpConnectionOverFlexRay

Containers included	
Container name	**Container definition**
XcpFlexrayBufferCfg_0	XcpFlexrayBufferCfg

Parameters included	
Parameter name	**Value**
XcpFlxNodeAddress	0
XcpFlxHeaderAlignment	PACKET_ALIGNMENT_8
XcpPackMultiMsgInOneFlexRayFrame	true
XcpMaxFlexMsgLengthInit	254

5.2.2.2.17. XcpFlexrayBufferCfg_0

Containers included	
Container name	**Container definition**
FLX_BUFCfg	FLX_BUFCfg
LPDU_IDCfg	LPDU_IDCfg

Parameters included	
Parameter name	**Value**

5.2.2.2.18. FLX_BUFCfg

Parameters included	
Parameter name	**Value**
XcpFlxBufId	0
XcpFlxBufMaxLen	VARIABLE_INITIALIZED
XcpFlxBufMaxLenInitValue	8

5.2.2.2.19. LPDU_IDCfg

Parameters included	
Parameter name	**Value**
XcpFlxSlotId	VARIABLE

Parameters included	
XcpFlxSlotIdInitValue	1
XcpFlxOffset	VARIABLE
XcpFlxOffsetInitValue	0
XcpFlxCycleRepetition	VARIABLE
XcpFlxCycleRepetitionInitValue	0
XcpFlxChannel	VARIABLE
XcpFlxChannelInitValue	A

5.2.2.2.20. XcpConnectionOverEthernet

Parameters included	
Parameter name	**Value**
XcpPackMultiMsgInOneEthernetFrame	false
XcpMaxEthernetMsgLengthInit	48
XcpAutoOpenSoCon	true

5.2.2.2.21. XcpTxPduConnectionInfo_0

Parameters included	
Parameter name	**Value**
XcpPduSupportForResErrCtoType	true
XcpPduSupportForEvServCtoType	false
XcpPduSupportForDaqDtoType	true
XcpPduSupportTxFromTxConfirmation	true
XcpDefaultStateDynamicTxPDU	**(DISABLED)**

5.2.2.2.22. XcpRxPduConnectionInfo_0

Parameters included	
Parameter name	**Value**
XcpRxLowerLayerHandleId	0
XcpPduSupportForCmdCtoType	true

Parameters included	
XcpPduSupportForStimDtoType	true
XcpDefaultStateDynamicRxPDU	**(DISABLED)**
XcpPduSupportRxFromRxIndication	true

5.2.2.2.23. XcpMemoryAccessArea

Parameters included	
Parameter name	**Value**
XcpMemoryAccessAreasSupport	false

5.2.2.2.24. XcpGeneral

Parameters included	
Parameter name	**Value**
XcpOnCanFDEnabled	false
XcpCalPagSupported	false
XcpDaqSupported	true
XcpStimSupported	true
XcpSlaveBlockModeSupported	true
XcpAddressGranularity	BYTE
XcpGranularityOdtEntrySizeDaq	BYTE
XcpGranularityOdtEntrySizeStim	BYTE
XcpMasterBlockModeSupported	true
XcpMaxBS	43
XcpMinST	200
XcpTimestampType	NO_TIME_STAMP
XcpDaqConfigType	DAQ_DYNAMIC
XcpPrescalerSupported	false
XcpStoreDaqSupported	false
XcpOdtEntrySizeDaq	1
XcpOdtEntrySizeStim	1
XcpOdtCount	4

Parameters included	
XcpIdentificationFieldType	ABSOLUTE
XcpTxRetryCount	0
XcpTimestampFixed	false
XcpTxBusRetry	0
XcpGetIdType1Callout	**(DISABLED)**
XcpGetIdType1MaxLength	**(DISABLED)**
XcpASAMMC2Filename	**(DISABLED)**
XcpDaqCount	1
XcpDevErrorDetect	false
XcpMainFunctionPeriod	0.02
XcpMaxCto	8
XcpMaxDto	8
XcpMaxEventChannel	0
XcpMinDaq	0
XcpOnCanEnabled	false
XcpOnCddEnabled	false
XcpOnEthernetEnabled	false
XcpOnFlexRayEnabled	true
XcpVersionInfoApi	false
XcpUserHeader	
XcpSeedAndKeyEnabled	false
XcpOdtEntriesCount	1
XcpSuppressTxSupport	false
XcpTimestampTicks	1000
XcpTimestampUnit	TIMESTAMP_UNIT_100MS
XcpCalPagStoreSupported	false
XcpUserTimestampSupported	false
XcpEventPacketEnabled	false
XcpWriteDaqMultipleSupported	false
XcpBuildChecksumSupport	false
XcpBuildChecksumType	XCP_CRC_16_CITT

Parameters included	
XcpBuildChecksumMainFunctionSupport	false
XcpCRCMainFunctionPeriod	
XcpAddressTranslationSupport	false
XcpModifyBitsSupported	false
XcpMaxCtoPgm	8
XcpEventTriggerMainFunc	false
XcpProcessMultiEventMainFunc	false
XcpDynamicAreaSize	1024
XcpDynamicMirrorAreaSize	1024
XcpPgmSupported	false
XcpMasterBlockModePgmSupported	false
XcpMaxBSPgm	1
XcpMinSTPgm	0
XcpTxBusTimeout	2.0
XcpBlockWriteReadMemoryRAMMechanism	false
XcpWriteMemoryRAMCallback	
XcpReadMemoryRAMCallback	
XcpEnableXcpControlApi	true
XcpDefaultXcpModuleState	true
XcpSupportForOverloadMSB	false
XcpSupportForGetDaqId	false
XcpSupportForFlxTPCommands	false
XcpSupportDynamicComChannels	false
XcpSupportComMainFunctions	**(DISABLED)**
XcpSupportForPDURoutingTable	false
XcpServiceWrapper	false

5.2.2.2.25. ReportToDem

Parameters included	
Parameter name	**Value**
XcpRetryFailedReportToDem	DISABLE

Parameters included	
XcpRetryFailedDemDetErrorId	30
XcpIllegalMemoryAccessReportToDem	DISABLE
XcpIllegalMemoryAccessDemDetErrorId	31
XcpPduBufferLockedReportToDem	DISABLE
XcpPduBufferLockedDemDetErrorId	130
XcpRespCTOQueueFullReportToDem	DISABLE
XcpRespCTOQueueFullDemDetErrorId	133
XcpPDUBufferFullReportToDem	DISABLE
XcpPDUBufferFullDemDetErrorId	129
XcpPDULostReportToDem	DISABLE
XcpPDULostDemDetErrorId	131
XcpStimulationDataLostReportToDem	DISABLE
XcpStimulationDataLostDemDetErrorId	135

5.2.2.2.26. XcpDefensiveProgramming

Parameters included	
Parameter name	**Value**
XcpDefProgEnabled	false
XcpPrecondAssertEnabled	false
XcpPostcondAssertEnabled	false
XcpStaticAssertEnabled	false
XcpUnreachAssertEnabled	false
XcpInvariantAssertEnabled	false

5.2.2.3. XcpRecConfigurationMinCan

Containers included	
Container name	**Container definition**
CommonPublishedInformation	CommonPublishedInformation
PublishedInformation	PublishedInformation
XcpConfig_0	XcpConfig

Containers included	
XcpGeneral	XcpGeneral
ReportToDem	ReportToDem
XcpDefensiveProgramming	XcpDefensiveProgramming

Parameters included	
Parameter name	**Value**
IMPLEMENTATION_CONFIG_VARIANT	VariantPreCompile

5.2.2.3.1. CommonPublishedInformation

Parameters included	
Parameter name	**Value**
ArMajorVersion	2
ArMinorVersion	0
ArPatchVersion	0
SwMajorVersion	2
SwMinorVersion	12
SwPatchVersion	2
ModuleId	212
VendorId	1
Release	

5.2.2.3.2. PublishedInformation

Parameters included	
Parameter name	**Value**
PbcfgMSupport	false

5.2.2.3.3. XcpConfig_0

Containers included	
Container name	**Container definition**
XcpDemEventParameterRefs	XcpDemEventParameterRefs

Containers included	
XcpEventChannel_0	XcpEventChannel
XcpPdu_Rx	XcpPdu
XcpPdu_Tx	XcpPdu
XcpPdu_RxBroadcast	XcpPdu
XcpConnectionCfg_0	XcpConnectionCfg
XcpMemoryAccessArea	XcpMemoryAccessArea

Parameters included	
Parameter name	**Value**

5.2.2.3.4. XcpDemEventParameterRefs

Parameters included	
Parameter name	**Value**

5.2.2.3.5. XcpEventChannel_0

Parameters included	
Parameter name	**Value**
XcpEventChannelConsistency	ODT
XcpEventChannelMaxDaqList	1
XcpEventChannelNumber	0
XcpEventChannelPriority	0
XcpEventChannelTimeCycle	1
XcpEventChannelTimeUnit	TIMESTAMP_UNIT_100MS
XcpEventChannelType	DAQ_STIM

5.2.2.3.6. XcpPdu_Rx

Containers included	
Container name	**Container definition**
XcpRxPdu	XcpRxPdu
XcpTxPdu	XcpTxPdu

Parameters included	
Parameter name	**Value**

5.2.2.3.7. XcpRxPdu

Parameters included	
Parameter name	**Value**
XcpRxPduId	0

5.2.2.3.8. XcpTxPdu

Parameters included	
Parameter name	**Value**
XcpTxPduId	0

5.2.2.3.9. XcpPdu_Tx

Containers included	
Container name	**Container definition**
XcpRxPdu	XcpRxPdu
XcpTxPdu	XcpTxPdu

Parameters included	
Parameter name	**Value**

5.2.2.3.10. XcpRxPdu

Parameters included	
Parameter name	**Value**
XcpRxPduId	1

5.2.2.3.11. XcpTxPdu

Parameters included	
Parameter name	**Value**

Parameters included	
XcpTxPduId	0

5.2.2.3.12. XcpPdu_RxBroadcast

Containers included	
Container name	**Container definition**
XcpRxPdu	XcpRxPdu
XcpTxPdu	XcpTxPdu

Parameters included	
Parameter name	**Value**

5.2.2.3.13. XcpRxPdu

Parameters included	
Parameter name	**Value**
XcpRxPduId	1

5.2.2.3.14. XcpTxPdu

Parameters included	
Parameter name	**Value**
XcpTxPduId	0

5.2.2.3.15. XcpConnectionCfg_0

Containers included	
Container name	**Container definition**
XcpTxPduConnectionInfo_0	XcpTxPduConnectionInfo
XcpRxPduConnectionInfo_0	XcpRxPduConnectionInfo
XcpRxPduConnectionInfo_1	XcpRxPduConnectionInfo

Parameters included	
Parameter name	**Value**

Parameters included	
XcpConnectionId	0

5.2.2.3.16. XcpConnectionInterfaceType

Containers included	
Container name	**Container definition**
XcpConnectionOverCAN	XcpConnectionInterfaceType XcpConnectionOverCAN
XcpConnectionOverCANFD	XcpConnectionInterfaceType XcpConnectionOver-CANFD
XcpConnectionOverFlexRay	XcpConnectionInterfaceType XcpConnectionOver-FlexRay
XcpConnectionOverEthernet	XcpConnectionInterfaceType XcpConnectionOverEther-net

Parameters included	
Parameter name	**Value**

5.2.2.3.17. XcpConnectionOverCAN

Parameters included	
Parameter name	**Value**

5.2.2.3.18. XcpConnectionOverCANFD

Parameters included	
Parameter name	**Value**
XcpCanFdMaxDlcRequired	
XcpCanFdMaxDlc	64
XcpCanFdFillValue	255

5.2.2.3.19. XcpConnectionOverFlexRay

Parameters included	
Parameter name	**Value**

Parameters included	
XcpFlxNodeAddress	0
XcpFlxHeaderAlignment	PACKET_ALIGNMENT_8
XcpPackMultiMsgInOneFlexRayFrame	false
XcpMaxFlexMsgLengthInit	254

5.2.2.3.20. XcpConnectionOverEthernet

Parameters included	
Parameter name	**Value**
XcpPackMultiMsgInOneEthernetFrame	false
XcpMaxEthernetMsgLengthInit	48
XcpAutoOpenSoCon	true

5.2.2.3.21. XcpTxPduConnectionInfo_0

Parameters included	
Parameter name	**Value**
XcpPduSupportForResErrCtoType	true
XcpPduSupportForEvServCtoType	true
XcpPduSupportForDaqDtoType	true
XcpPduSupportTxFromTxConfirmation	true
XcpDefaultStateDynamicTxPDU	**(DISABLED)**

5.2.2.3.22. XcpRxPduConnectionInfo_0

Parameters included	
Parameter name	**Value**
XcpRxLowerLayerHandleId	0
XcpPduSupportForCmdCtoType	true
XcpPduSupportForStimDtoType	true
XcpDefaultStateDynamicRxPDU	**(DISABLED)**
XcpPduSupportRxFromRxIndication	false

5.2.2.3.23. XcpRxPduConnectionInfo_1

Parameters included	
Parameter name	**Value**
XcpRxLowerLayerHandleId	0
XcpPduSupportForCmdCtoType	true
XcpPduSupportForStimDtoType	false
XcpDefaultStateDynamicRxPDU	**(DISABLED)**
XcpPduSupportRxFromRxIndication	false

5.2.2.3.24. XcpMemoryAccessArea

Parameters included	
Parameter name	**Value**
XcpMemoryAccessAreasSupport	false

5.2.2.3.25. XcpGeneral

Parameters included	
Parameter name	**Value**
XcpOnCanFDEnabled	false
XcpCalPagSupported	false
XcpDaqSupported	true
XcpStimSupported	true
XcpSlaveBlockModeSupported	true
XcpAddressGranularity	BYTE
XcpGranularityOdtEntrySizeDaq	BYTE
XcpGranularityOdtEntrySizeStim	BYTE
XcpMasterBlockModeSupported	true
XcpMaxBS	43
XcpMinST	200
XcpTimestampType	NO_TIME_STAMP
XcpDaqConfigType	DAQ_DYNAMIC
XcpPrescalerSupported	false

Parameters included	
XcpStoreDaqSupported	false
XcpOdtEntrySizeDaq	1
XcpOdtEntrySizeStim	1
XcpOdtCount	4
XcpIdentificationFieldType	ABSOLUTE
XcpTxRetryCount	0
XcpTimestampFixed	false
XcpTxBusRetry	0
XcpGetIdType1Callout	**(DISABLED)**
XcpGetIdType1MaxLength	**(DISABLED)**
XcpASAMMC2Filename	**(DISABLED)**
XcpDaqCount	2
XcpDevErrorDetect	false
XcpMainFunctionPeriod	0.02
XcpMaxCto	8
XcpMaxDto	8
XcpMaxEventChannel	0
XcpMinDaq	0
XcpOnCanEnabled	true
XcpOnCddEnabled	false
XcpOnEthernetEnabled	false
XcpOnFlexRayEnabled	false
XcpVersionInfoApi	false
XcpUserHeader	
XcpSeedAndKeyEnabled	false
XcpOdtEntriesCount	7
XcpSuppressTxSupport	false
XcpTimestampTicks	1000
XcpTimestampUnit	TIMESTAMP_UNIT_100MS
XcpCalPagStoreSupported	false
XcpUserTimestampSupported	false

Parameters included	
XcpEventPacketEnabled	true
XcpWriteDaqMultipleSupported	false
XcpBuildChecksumSupport	true
XcpBuildChecksumType	XCP_CRC_16_CITT
XcpBuildChecksumMainFunctionSupport	false
XcpCRCMainFunctionPeriod	
XcpAddressTranslationSupport	false
XcpModifyBitsSupported	false
XcpMaxCtoPgm	8
XcpEventTriggerMainFunc	false
XcpProcessMultiEventMainFunc	false
XcpDynamicAreaSize	1024
XcpDynamicMirrorAreaSize	1024
XcpPgmSupported	true
XcpMasterBlockModePgmSupported	true
XcpMaxBSPgm	43
XcpMinSTPgm	200
XcpTxBusTimeout	2.0
XcpBlockWriteReadMemoryRAMMechanism	false
XcpWriteMemoryRAMCallback	
XcpReadMemoryRAMCallback	
XcpEnableXcpControlApi	true
XcpDefaultXcpModuleState	true
XcpSupportForOverloadMSB	false
XcpSupportForGetDaqId	false
XcpSupportForFlxTPCommands	false
XcpSupportDynamicComChannels	false
XcpSupportComMainFunctions	**(DISABLED)**
XcpSupportForPDURoutingTable	false
XcpServiceWrapper	false

5.2.2.3.26. ReportToDem

Parameters included	
Parameter name	**Value**
XcpRetryFailedReportToDem	DISABLE
XcpRetryFailedDemDetErrorId	30
XcpIllegalMemoryAccessReportToDem	DISABLE
XcpIllegalMemoryAccessDemDetErrorId	31
XcpPduBufferLockedReportToDem	DISABLE
XcpPduBufferLockedDemDetErrorId	130
XcpRespCTOQueueFullReportToDem	DISABLE
XcpRespCTOQueueFullDemDetErrorId	133
XcpPDUBufferFullReportToDem	DISABLE
XcpPDUBufferFullDemDetErrorId	129
XcpPDULostReportToDem	DISABLE
XcpPDULostDemDetErrorId	131
XcpStimulationDataLostReportToDem	DISABLE
XcpStimulationDataLostDemDetErrorId	135

5.2.2.3.27. XcpDefensiveProgramming

Parameters included	
Parameter name	**Value**
XcpDefProgEnabled	false
XcpPrecondAssertEnabled	false
XcpPostcondAssertEnabled	false
XcpStaticAssertEnabled	false
XcpUnreachAssertEnabled	false
XcpInvariantAssertEnabled	false

5.2.2.4. XcpRecConfigurationMinEthernetTCP

Containers included	
Container name	**Container definition**
CommonPublishedInformation	CommonPublishedInformation
PublishedInformation	PublishedInformation
XcpConfig_0	XcpConfig
XcpGeneral	XcpGeneral
ReportToDem	ReportToDem
XcpDefensiveProgramming	XcpDefensiveProgramming

Parameters included	
Parameter name	**Value**
IMPLEMENTATION_CONFIG_VARIANT	VariantPreCompile

5.2.2.4.1. CommonPublishedInformation

Parameters included	
Parameter name	**Value**
ArMajorVersion	2
ArMinorVersion	0
ArPatchVersion	0
SwMajorVersion	2
SwMinorVersion	12
SwPatchVersion	2
ModuleId	212
VendorId	1
Release	

5.2.2.4.2. PublishedInformation

Parameters included	
Parameter name	**Value**
PbcfgMSupport	false

5.2.2.4.3. XcpConfig_0

Containers included	
Container name	**Container definition**
XcpDaqList_0	XcpDaqList
XcpDaqList_1	XcpDaqList
XcpDaqList_2	XcpDaqList
XcpDaqList_3	XcpDaqList
XcpDemEventParameterRefs	XcpDemEventParameterRefs
XcpEventChannel_0	XcpEventChannel
XcpPdu_Rx	XcpPdu
XcpPdu_Tx	XcpPdu
XcpConnectionCfg_0	XcpConnectionCfg
XcpMemoryAccessArea	XcpMemoryAccessArea

Parameters included	
Parameter name	**Value**

5.2.2.4.4. XcpDaqList_0

Parameters included	
Parameter name	**Value**
XcpDaqListNumber	0
XcpDaqListType	DAQ
XcpMaxOdt	1
XcpMaxOdtEntries	1
XcpDaqEventFixed	false
XcpDaqListPriority	0
XcpDaqListPrescaler	1

5.2.2.4.5. XcpDaqList_1

Parameters included	
Parameter name	**Value**

Parameters included	
XcpDaqListNumber	1
XcpDaqListType	DAQ_STIM
XcpMaxOdt	1
XcpMaxOdtEntries	1
XcpDaqEventFixed	false
XcpDaqListPriority	0
XcpDaqListPrescaler	1

5.2.2.4.6. XcpDaqList_2

Parameters included	
Parameter name	**Value**
XcpDaqListNumber	2
XcpDaqListType	STIM
XcpMaxOdt	1
XcpMaxOdtEntries	1
XcpDaqEventFixed	false
XcpDaqListPriority	0
XcpDaqListPrescaler	1

5.2.2.4.7. XcpDaqList_3

Parameters included	
Parameter name	**Value**
XcpDaqListNumber	3
XcpDaqListType	DAQ_STIM
XcpMaxOdt	1
XcpMaxOdtEntries	1
XcpDaqEventFixed	false
XcpDaqListPriority	0
XcpDaqListPrescaler	1

5.2.2.4.8. XcpDemEventParameterRefs

Parameters included	
Parameter name	**Value**

5.2.2.4.9. XcpEventChannel_0

Parameters included	
Parameter name	**Value**
XcpEventChannelConsistency	ODT
XcpEventChannelMaxDaqList	1
XcpEventChannelNumber	0
XcpEventChannelPriority	0
XcpEventChannelTimeCycle	1
XcpEventChannelTimeUnit	TIMESTAMP_UNIT_1S
XcpEventChannelType	DAQ_STIM

5.2.2.4.10. XcpPdu_Rx

Containers included	
Container name	**Container definition**
XcpRxPdu	XcpRxPdu
XcpTxPdu	XcpTxPdu

Parameters included	
Parameter name	**Value**

5.2.2.4.11. XcpRxPdu

Parameters included	
Parameter name	**Value**
XcpRxPduId	0

5.2.2.4.12. XcpTxPdu

Parameters included	
Parameter name	**Value**
XcpTxPduId	0

5.2.2.4.13. XcpPdu_Tx

Containers included	
Container name	**Container definition**
XcpRxPdu	XcpRxPdu
XcpTxPdu	XcpTxPdu

Parameters included	
Parameter name	**Value**

5.2.2.4.14. XcpRxPdu

Parameters included	
Parameter name	**Value**
XcpRxPduId	1

5.2.2.4.15. XcpTxPdu

Parameters included	
Parameter name	**Value**
XcpTxPduId	0

5.2.2.4.16. XcpConnectionCfg_0

Containers included	
Container name	**Container definition**
XcpTxPduConnectionInfo_0	XcpTxPduConnectionInfo

Containers included	
XcpRxPduConnectionInfo_0	XcpRxPduConnectionInfo

Parameters included	
Parameter name	**Value**
XcpConnectionId	0

5.2.2.4.17. XcpConnectionInterfaceType

Containers included	
Container name	**Container definition**
XcpConnectionOverCAN	XcpConnectionInterfaceType XcpConnectionOverCAN
XcpConnectionOverCANFD	XcpConnectionInterfaceType XcpConnectionOver-CANFD
XcpConnectionOverFlexRay	XcpConnectionInterfaceType XcpConnectionOver-FlexRay
XcpConnectionOverEthernet	XcpConnectionInterfaceType XcpConnectionOverEther-net

Parameters included	
Parameter name	**Value**

5.2.2.4.18. XcpConnectionOverCAN

Parameters included	
Parameter name	**Value**

5.2.2.4.19. XcpConnectionOverCANFD

Parameters included	
Parameter name	**Value**
XcpCanFdMaxDlcRequired	
XcpCanFdMaxDlc	64
XcpCanFdFillValue	255

5.2.2.4.20. XcpConnectionOverFlexRay

Parameters included	
Parameter name	**Value**
XcpFlxNodeAddress	0
XcpFlxHeaderAlignment	PACKET_ALIGNMENT_8
XcpPackMultiMsgInOneFlexRayFrame	false
XcpMaxFlexMsgLengthInit	254

5.2.2.4.21. XcpConnectionOverEthernet

Parameters included	
Parameter name	**Value**
XcpPackMultiMsgInOneEthernetFrame	false
XcpMaxEthernetMsgLengthInit	48
XcpAutoOpenSoCon	false

5.2.2.4.22. XcpTxPduConnectionInfo_0

Parameters included	
Parameter name	**Value**
XcpPduSupportForResErrCtoType	true
XcpPduSupportForEvServCtoType	false
XcpPduSupportForDaqDtoType	true
XcpPduSupportTxFromTxConfirmation	true
XcpDefaultStateDynamicTxPDU	**(DISABLED)**

5.2.2.4.23. XcpRxPduConnectionInfo_0

Parameters included	
Parameter name	**Value**
XcpRxLowerLayerHandleId	0
XcpPduSupportForCmdCtoType	true
XcpPduSupportForStimDtoType	true

Parameters included	
XcpDefaultStateDynamicRxPDU	**(DISABLED)**
XcpPduSupportRxFromRxIndication	true

5.2.2.4.24. XcpMemoryAccessArea

Parameters included	
Parameter name	**Value**
XcpMemoryAccessAreasSupport	false

5.2.2.4.25. XcpGeneral

Parameters included	
Parameter name	**Value**
XcpOnCanFDEnabled	false
XcpCalPagSupported	false
XcpDaqSupported	true
XcpStimSupported	true
XcpSlaveBlockModeSupported	true
XcpAddressGranularity	BYTE
XcpMasterBlockModeSupported	true
XcpMaxBS	2
XcpMinST	200
XcpTimestampType	NO_TIME_STAMP
XcpDaqConfigType	DAQ_DYNAMIC
XcpPrescalerSupported	false
XcpStoreDaqSupported	false
XcpOdtEntrySizeDaq	255
XcpOdtEntrySizeStim	255
XcpOdtCount	4
XcpIdentificationFieldType	ABSOLUTE
XcpTxRetryCount	0
XcpTimestampFixed	false

Parameters included	
XcpTxBusRetry	0
XcpGetIdType1Callout	**(DISABLED)**
XcpGetIdType1MaxLength	**(DISABLED)**
XcpASAMMC2Filename	**(DISABLED)**
XcpDaqCount	1
XcpDevErrorDetect	false
XcpMainFunctionPeriod	0.02
XcpMaxCto	255
XcpMaxDto	1024
XcpMaxEventChannel	0
XcpMinDaq	0
XcpOnCanEnabled	false
XcpOnCddEnabled	false
XcpOnEthernetEnabled	true
XcpOnFlexRayEnabled	false
XcpVersionInfoApi	false
XcpUserHeader	
XcpSeedAndKeyEnabled	false
XcpOdtEntriesCount	1
XcpSuppressTxSupport	false
XcpTimestampTicks	1000
XcpTimestampUnit	TIMESTAMP_UNIT_100MS
XcpCalPagStoreSupported	false
XcpUserTimestampSupported	false
XcpEventPacketEnabled	false
XcpWriteDaqMultipleSupported	false
XcpBuildChecksumSupport	false
XcpBuildChecksumType	XCP_CRC_16_CITT
XcpBuildChecksumMainFunctionSupport	false
XcpCRCMainFunctionPeriod	
XcpAddressTranslationSupport	false

Parameters included	
XcpModifyBitsSupported	false
XcpMaxCtoPgm	8
XcpEventTriggerMainFunc	false
XcpProcessMultiEventMainFunc	false
XcpDynamicAreaSize	1024
XcpDynamicMirrorAreaSize	1024
XcpPgmSupported	false
XcpMasterBlockModePgmSupported	false
XcpMaxBSPgm	1
XcpMinSTPgm	0
XcpTxBusTimeout	2.0
XcpBlockWriteReadMemoryRAMMechanism	false
XcpWriteMemoryRAMCallback	
XcpReadMemoryRAMCallback	
XcpEnableXcpControlApi	true
XcpDefaultXcpModuleState	true
XcpSupportForOverloadMSB	false
XcpSupportForGetDaqId	false
XcpSupportForFlxTPCommands	false
XcpSupportDynamicComChannels	false
XcpSupportComMainFunctions	**(DISABLED)**
XcpSupportForPDURoutingTable	false
XcpServiceWrapper	false

5.2.2.4.26. ReportToDem

Parameters included	
Parameter name	**Value**
XcpRetryFailedReportToDem	DISABLE
XcpRetryFailedDemDetErrorId	30
XcpIllegalMemoryAccessReportToDem	DISABLE
XcpIllegalMemoryAccessDemDetErrorId	30

Parameters included	
XcpPduBufferLockedReportToDem	DISABLE
XcpPduBufferLockedDemDetErrorId	130
XcpRespCTOQueueFullReportToDem	DISABLE
XcpRespCTOQueueFullDemDetErrorId	133
XcpPDUBufferFullReportToDem	DISABLE
XcpPDUBufferFullDemDetErrorId	129
XcpPDULostReportToDem	DISABLE
XcpPDULostDemDetErrorId	131
XcpStimulationDataLostReportToDem	DISABLE
XcpStimulationDataLostDemDetErrorId	135

5.2.2.4.27. XcpDefensiveProgramming

Parameters included	
Parameter name	**Value**
XcpDefProgEnabled	false
XcpPrecondAssertEnabled	false
XcpPostcondAssertEnabled	false
XcpStaticAssertEnabled	false
XcpUnreachAssertEnabled	false
XcpInvariantAssertEnabled	false

5.2.2.5. XcpRecConfigurationMinEthernetUDP

Containers included	
Container name	**Container definition**
CommonPublishedInformation	CommonPublishedInformation
PublishedInformation	PublishedInformation
XcpConfig_0	XcpConfig
XcpGeneral	XcpGeneral
ReportToDem	ReportToDem
XcpDefensiveProgramming	XcpDefensiveProgramming

Parameters included	
Parameter name	**Value**
IMPLEMENTATION_CONFIG_VARIANT	VariantPreCompile

5.2.2.5.1. CommonPublishedInformation

Parameters included	
Parameter name	**Value**
ArMajorVersion	2
ArMinorVersion	0
ArPatchVersion	0
SwMajorVersion	2
SwMinorVersion	12
SwPatchVersion	2
ModuleId	212
VendorId	1
Release	

5.2.2.5.2. PublishedInformation

Parameters included	
Parameter name	**Value**
PbcfgMSupport	false

5.2.2.5.3. XcpConfig_0

Containers included	
Container name	**Container definition**
XcpDaqList_0	XcpDaqList
XcpDaqList_1	XcpDaqList
XcpDaqList_2	XcpDaqList
XcpDaqList_3	XcpDaqList
XcpDemEventParameterRefs	XcpDemEventParameterRefs

Containers included	
XcpEventChannel_0	XcpEventChannel
XcpPdu_Rx	XcpPdu
XcpPdu_Tx	XcpPdu
XcpConnectionCfg_0	XcpConnectionCfg
XcpMemoryAccessArea	XcpMemoryAccessArea

Parameters included	
Parameter name	**Value**

5.2.2.5.4. XcpDaqList_0

Parameters included	
Parameter name	**Value**
XcpDaqListNumber	0
XcpDaqListType	DAQ
XcpMaxOdt	1
XcpMaxOdtEntries	1
XcpDaqEventFixed	false
XcpDaqListPriority	0
XcpDaqListPrescaler	1

5.2.2.5.5. XcpDaqList_1

Parameters included	
Parameter name	**Value**
XcpDaqListNumber	1
XcpDaqListType	DAQ_STIM
XcpMaxOdt	1
XcpMaxOdtEntries	1
XcpDaqEventFixed	false
XcpDaqListPriority	0
XcpDaqListPrescaler	1

5.2.2.5.6. XcpDaqList_2

Parameters included	
Parameter name	**Value**
XcpDaqListNumber	2
XcpDaqListType	STIM
XcpMaxOdt	1
XcpMaxOdtEntries	1
XcpDaqEventFixed	false
XcpDaqListPriority	0
XcpDaqListPrescaler	1

5.2.2.5.7. XcpDaqList_3

Parameters included	
Parameter name	**Value**
XcpDaqListNumber	3
XcpDaqListType	DAQ
XcpMaxOdt	1
XcpMaxOdtEntries	1
XcpDaqEventFixed	false
XcpDaqListPriority	0
XcpDaqListPrescaler	1

5.2.2.5.8. XcpDemEventParameterRefs

Parameters included	
Parameter name	**Value**

5.2.2.5.9. XcpEventChannel_0

Parameters included	
Parameter name	**Value**
XcpEventChannelConsistency	ODT

Parameters included	
XcpEventChannelMaxDaqList	1
XcpEventChannelNumber	0
XcpEventChannelPriority	0
XcpEventChannelTimeCycle	1
XcpEventChannelTimeUnit	TIMESTAMP_UNIT_1S
XcpEventChannelType	DAQ_STIM

5.2.2.5.10. XcpPdu_Rx

Containers included	
Container name	**Container definition**
XcpRxPdu	XcpRxPdu
XcpTxPdu	XcpTxPdu

Parameters included	
Parameter name	**Value**

5.2.2.5.11. XcpRxPdu

Parameters included	
Parameter name	**Value**
XcpRxPduId	0

5.2.2.5.12. XcpTxPdu

Parameters included	
Parameter name	**Value**
XcpTxPduId	0

5.2.2.5.13. XcpPdu_Tx

Containers included	
Container name	**Container definition**

Containers included	
XcpRxPdu	XcpRxPdu
XcpTxPdu	XcpTxPdu

Parameters included	
Parameter name	**Value**

5.2.2.5.14. XcpRxPdu

Parameters included	
Parameter name	**Value**
XcpRxPduId	1

5.2.2.5.15. XcpTxPdu

Parameters included	
Parameter name	**Value**
XcpTxPduId	0

5.2.2.5.16. XcpConnectionCfg_0

Containers included	
Container name	**Container definition**
XcpTxPduConnectionInfo_0	XcpTxPduConnectionInfo
XcpRxPduConnectionInfo_0	XcpRxPduConnectionInfo

Parameters included	
Parameter name	**Value**
XcpConnectionId	0

5.2.2.5.17. XcpConnectionInterfaceType

Containers included	
Container name	**Container definition**

Containers included	
XcpConnectionOverCAN	XcpConnectionInterfaceType XcpConnectionOverCAN
XcpConnectionOverCANFD	XcpConnectionInterfaceType XcpConnectionOver-CANFD
XcpConnectionOverFlexRay	XcpConnectionInterfaceType XcpConnectionOver-FlexRay
XcpConnectionOverEthernet	XcpConnectionInterfaceType XcpConnectionOverEther-net

Parameters included	
Parameter name	**Value**

5.2.2.5.18. XcpConnectionOverCAN

Parameters included	
Parameter name	**Value**

5.2.2.5.19. XcpConnectionOverCANFD

Parameters included	
Parameter name	**Value**
XcpCanFdMaxDlcRequired	
XcpCanFdMaxDlc	64
XcpCanFdFillValue	255

5.2.2.5.20. XcpConnectionOverFlexRay

Parameters included	
Parameter name	**Value**
XcpFlxNodeAddress	0
XcpFlxHeaderAlignment	PACKET_ALIGNMENT_8
XcpPackMultiMsgInOneFlexRayFrame	false
XcpMaxFlexMsgLengthInit	254

5.2.2.5.21. XcpConnectionOverEthernet

Parameters included	
Parameter name	**Value**
XcpPackMultiMsgInOneEthernetFrame	false
XcpMaxEthernetMsgLengthInit	48
XcpAutoOpenSoCon	true

5.2.2.5.22. XcpTxPduConnectionInfo_0

Parameters included	
Parameter name	**Value**
XcpPduSupportForResErrCtoType	true
XcpPduSupportForEvServCtoType	false
XcpPduSupportForDaqDtoType	true
XcpPduSupportTxFromTxConfirmation	true
XcpDefaultStateDynamicTxPDU	**(DISABLED)**

5.2.2.5.23. XcpRxPduConnectionInfo_0

Parameters included	
Parameter name	**Value**
XcpRxLowerLayerHandleId	0
XcpPduSupportForCmdCtoType	true
XcpPduSupportForStimDtoType	true
XcpDefaultStateDynamicRxPDU	**(DISABLED)**
XcpPduSupportRxFromRxIndication	true

5.2.2.5.24. XcpMemoryAccessArea

Parameters included	
Parameter name	**Value**
XcpMemoryAccessAreasSupport	false

5.2.2.5.25. XcpGeneral

Parameters included	
Parameter name	**Value**
XcpOnCanFDEnabled	false
XcpCalPagSupported	false
XcpDaqSupported	true
XcpStimSupported	true
XcpSlaveBlockModeSupported	true
XcpAddressGranularity	BYTE
XcpMasterBlockModeSupported	true
XcpMaxBS	43
XcpMinST	200
XcpTimestampType	NO_TIME_STAMP
XcpDaqConfigType	DAQ_DYNAMIC
XcpPrescalerSupported	false
XcpStoreDaqSupported	false
XcpOdtEntrySizeDaq	1
XcpOdtEntrySizeStim	1
XcpOdtCount	1
XcpIdentificationFieldType	ABSOLUTE
XcpTxRetryCount	0
XcpTimestampFixed	false
XcpTxBusRetry	0
XcpGetIdType1Callout	**(DISABLED)**
XcpGetIdType1MaxLength	**(DISABLED)**
XcpASAMMC2Filename	**(DISABLED)**
XcpDaqCount	4
XcpDevErrorDetect	false
XcpMainFunctionPeriod	0.02
XcpMaxCto	8
XcpMaxDto	8
XcpMaxEventChannel	0

Parameters included	
XcpMinDaq	4
XcpOnCanEnabled	false
XcpOnCddEnabled	false
XcpOnEthernetEnabled	true
XcpOnFlexRayEnabled	false
XcpVersionInfoApi	false
XcpUserHeader	
XcpSeedAndKeyEnabled	false
XcpOdtEntriesCount	1
XcpSuppressTxSupport	false
XcpTimestampTicks	1000
XcpTimestampUnit	TIMESTAMP_UNIT_100MS
XcpCalPagStoreSupported	false
XcpUserTimestampSupported	false
XcpEventPacketEnabled	false
XcpWriteDaqMultipleSupported	false
XcpBuildChecksumSupport	false
XcpBuildChecksumType	XCP_CRC_16_CITT
XcpBuildChecksumMainFunctionSupport	false
XcpCRCMainFunctionPeriod	
XcpAddressTranslationSupport	false
XcpModifyBitsSupported	false
XcpMaxCtoPgm	8
XcpEventTriggerMainFunc	false
XcpProcessMultiEventMainFunc	false
XcpDynamicAreaSize	1024
XcpDynamicMirrorAreaSize	1024
XcpPgmSupported	false
XcpMasterBlockModePgmSupported	false
XcpMaxBSPgm	1
XcpMinSTPgm	0

Parameters included	
XcpTxBusTimeout	2.0
XcpBlockWriteReadMemoryRAMMechanism	false
XcpWriteMemoryRAMCallback	
XcpReadMemoryRAMCallback	
XcpEnableXcpControlApi	true
XcpDefaultXcpModuleState	true
XcpSupportForOverloadMSB	false
XcpSupportForGetDaqId	false
XcpSupportForFlxTPCommands	false
XcpSupportDynamicComChannels	false
XcpSupportComMainFunctions	**(DISABLED)**
XcpSupportForPDURoutingTable	false
XcpServiceWrapper	false

5.2.2.5.26. ReportToDem

Parameters included	
Parameter name	**Value**
XcpRetryFailedReportToDem	DISABLE
XcpRetryFailedDemDetErrorId	30
XcpIllegalMemoryAccessReportToDem	DISABLE
XcpIllegalMemoryAccessDemDetErrorId	31
XcpPduBufferLockedReportToDem	DISABLE
XcpPduBufferLockedDemDetErrorId	130
XcpRespCTOQueueFullReportToDem	DISABLE
XcpRespCTOQueueFullDemDetErrorId	133
XcpPDUBufferFullReportToDem	DISABLE
XcpPDUBufferFullDemDetErrorId	129
XcpPDULostReportToDem	DISABLE
XcpPDULostDemDetErrorId	131
XcpStimulationDataLostReportToDem	DISABLE
XcpStimulationDataLostDemDetErrorId	135

5.2.2.5.27. XcpDefensiveProgramming

Parameters included	
Parameter name	**Value**
XcpDefProgEnabled	false
XcpPrecondAssertEnabled	false
XcpPostcondAssertEnabled	false
XcpStaticAssertEnabled	false
XcpUnreachAssertEnabled	false
XcpInvariantAssertEnabled	false

5.2.3. Application programming interface (API)

5.2.3.1. Type definitions

5.2.3.1.1. Xcp_ApplReturnType

Purpose	XCP application callout return type.
Type	`uint8`
Description	This type provides the different return values to be used by the application callouts.

5.2.3.1.2. Xcp_ConfigType

Purpose	Type for the configuration data (place holder because of pre-compile time configuration support).	
Type	struct	
Members	`uint32 Dummy`	Dummy variable to have valid C code

5.2.3.1.3. Xcp_ReturnType

Purpose	Xcp return type for EB specific API functions.

Type	uint8

5.2.3.1.4. Xcp_TransmissionModeType

Purpose	Xcp_Transmission Mode Type: handles the enabling and disabling of the transmission mode Possible values can be XCP_TX_OFF and XCP_TX_ON.
Type	uint8

5.2.3.2. Macro constants

5.2.3.2.1. XCP_APPL_ERR_ACCESS_DENIED

Purpose	return value of XCP application callout with XCP ERR_ACCESS_DENIED error code.
Value	0x24U

5.2.3.2.2. XCP_APPL_ERR_CMD_UNKNOWN

Purpose	return value of XCP application callout with XCP ERR_CMD_UNKNOWN error code.
Value	0x20U

5.2.3.2.3. XCP_APPL_ERR_GENERIC

Purpose	return value of XCP application callout with XCP ERR_GENERIC error code.
Value	0x31U

5.2.3.2.4. XCP_APPL_ERR_MEMORY_OVERFLOW

Purpose	return value of XCP application callout with XCP ERR_MEMORY_OVERFLOW error code.
Value	0x30U

5.2.3.2.5. XCP_APPL_ERR_MODE_NOT_VALID

Purpose	return value of XCP application callout with XCP ERR_MODE_NOT_VALID error code.
Value	0x27U

5.2.3.2.6. XCP_APPL_ERR_OUT_OF_RANGE

Purpose	return value of XCP application callout with XCP ERR_OUT_OF_RANGE error code.
Value	0x22U

5.2.3.2.7. XCP_APPL_ERR_PAGE_NOT_VALID

Purpose	return value of XCP application callout with XCP ERR_PAGE_NOT_VALID error code.
Value	0x26U

5.2.3.2.8. XCP_APPL_ERR_SEGMENT_NOT_VALID

Purpose	return value of XCP application callout XCP ERR_SEGMENT_NOT_VALID error code.
Value	0x28U

5.2.3.2.9. XCP_APPL_ERR_WRITE_PROTECTED

Purpose	return value of XCP application callout with XCP ERR_WRITE_PROTECTED error code.
Value	0x23U

5.2.3.2.10. XCP_APPL_OK

Purpose	return value of XCP application callout for successful execution.
Value	0x01U

5.2.3.2.11. XCP_AR_RELEASE_MAJOR_VERSION

Purpose	AUTOSAR release major version.
Value	4U

5.2.3.2.12. XCP_AR_RELEASE_MINOR_VERSION

Purpose	AUTOSAR release minor version.
Value	0U

5.2.3.2.13. XCP_AR_RELEASE_REVISION_VERSION

Purpose	AUTOSAR release revision version.
Value	3U

5.2.3.2.14. XCP_ERR_STIMULATION_DATA_LOST

Purpose	Error Code.
Value	0x87U
Description	The error is triggered by Xcp_InsertOdtToStimBuffer() because previous stimulation data was used (it exists) and therefore a new entry cannot be added. Consequently the new data is lost and a DET notification shall be issued to signal this situation.

5.2.3.2.15. XCP_ERR_TX_BUFFER_NOT_AVAILABLE

Purpose	Error Code.
Value	0x88U
Description	The error is triggered by Xcp_UpdateAvailableTxBuffers() because the TX buffer is not available and thus there is the great possibility to loose Pdu data.

5.2.3.2.16. XCP_EVENT_CTO_QUEUE_FULL

Purpose	Error Code.

Value	0x85U
Description	The error is triggered by Xcp_PrepareStoredDaqListsResume() if the event CTO queue, used to enqueue the event CTO, is full. Therefore the incoming event CTO shall not be enqueued and this matter should be signaled.

5.2.3.2.17. XCP_E_DATA_LOST

Purpose	Error Code thrown when data which should be provided from other partition is lost.
Value	0x93U

5.2.3.2.18. XCP_E_INVALID_EVENT

Purpose	Error Code.
Value	0x40U
Description	The error is triggered by Xcp_SetEvent() if ► The given event ID (parameter) is invalid, i.e. there is no event with that ID or ► The given event ID belongs to a periodic event, for which this function shall not be called.

5.2.3.2.19. XCP_E_INVALID_NVM_BLOCK_LENGTH

Purpose	Error Code thrown when the NvM block needed to store the DAQ lists has an invalid length.
Value	0x20U
Description	The storage of DAQ lists in NV memory requires a NvM block to be configured correctly. This error describes the situation when the block required by XCP to store the DAQ list configuration has an invalid length. The length of the NvM block must match the one of the runtime structure of the DAQ lists. In order to determine the correct length, the easiest way is to check with the debugger the size of the runtime DAQ lists information (hint: breakpoint where this error is thrown).

5.2.3.2.20. XCP_E_INVALID_PDUID

Purpose	Error Code.

Value	0x03U
Description	API called with wrong PDU ID.

5.2.3.2.21. XCP_E_INVALID_RX_PDU_LENGTH

Purpose	Error Code.
Value	0x80U
Description	The error is triggered by Xcp_<If>RxIndication() if the length of the received Pdu is greater than maximum configured buffer size for the incoming PDUs

5.2.3.2.22. XCP_E_INVALID_STATE_COMMAND

Purpose	Error Code thrown when the XCP Control function receives an invalid command. This error describes the situation when the function is called with a value other then 0xAA or 0x55.
Value	0x30U

5.2.3.2.23. XCP_E_INV_POINTER

Purpose	Error Code.
Value	0x01U
Description	API is invoked with invalid pointer.

5.2.3.2.24. XCP_E_MEMORY_PROXY_CYCLIC_EVENT_SYNC_ERROR

Purpose	Error Code thrown when the XCP sets the cyclic event but the processing of the event could not be processed in the previous MFs.
Value	0x92U

5.2.3.2.25. XCP_E_MEMORY_PROXY_MF_SYNC_ERROR

Purpose	Error Code thrown when the XCP has to trigger the processing of an event if the event could not be processed in the previous MF.

Value	0x91U

5.2.3.2.26. XCP_E_NOT_INITIALIZED

Purpose	Error Code.
Value	0x02U
Description	API is called before complete initialization.

5.2.3.2.27. XCP_E_NULL_POINTER

Purpose	Error Code.
Value	0x12U
Description	Null pointer has been passed as an argument.

5.2.3.2.28. XCP_E_PARAM_CHANNEL_INVALID

Purpose	Error Code.
Value	0x89U
Description	The error is triggered by Xcp_SetTransmissionMode() when the channel parameter is invalid

5.2.3.2.29. XCP_E_PDU_BUFFER_FULL

Purpose	Error Code.
Value	0x81U
Description	The error is triggered by Xcp_<If>RxIndication() if the L-PDU buffer corresponding to the received PDU is full. This may happen when the Xcp_MainFunction is called with higher latency then the lowest latency used by the master to send messages.

5.2.3.2.30. XCP_E_PDU_BUFFER_LOCKED

Purpose	Error Code.
Value	0x82U

Description	The error is triggered by Xcp_<If>(), in case the PDU doesn't belong the current connection.

5.2.3.2.31. XCP_E_PDU_LOST

Purpose	Error Code.
Value	0x83U
Description	The error is triggered by Xcp_MainFunction()/Xcp_MainFunction_[EventName]/ Xcp_<If>RxIndication() if, when unpacking PDUs, at least one PDU message is silently lost, i.e. without notification towards the Xcp master.

5.2.3.2.32. XCP_E_SCHM_ERROR

Purpose	Error Code.
Value	0X90U
Description	The error is triggered by Xcp_MainFunction in the BSW distribution scenario when a memory access (read,write) requires the usage of a memory proxy which cannot be triggered due to a SchM problem

5.2.3.2.33. XCP_E_WRONG_CONTEXT

Purpose	Error Code.
Value	0x92U
Description	The API has been called from an invalid application context.

5.2.3.2.34. XCP_INSTANCE_ID

Purpose	Id of instance of Xcp.
Value	0U

5.2.3.2.35. XCP_MODULE_ID

Purpose	AUTOSAR module identification.

Value	212U

5.2.3.2.36. XCP_NOT_CONNECTED

Purpose	Xcp EB specific API return value when Xcp slave is not connected to the master.
Value	3U

5.2.3.2.37. XCP_NOT_INITIALIZED

Purpose	Xcp EB specific API return value when Xcp module is not initialized.
Value	2U

5.2.3.2.38. XCP_NOT_OK

Purpose	Xcp EB specific API return value when development error happened.
Value	1U

5.2.3.2.39. XCP_NO_ACTIVE_LIST

Purpose	Xcp EB specific API return value when no active list associated (Xcp_SetEvent()).
Value	5U

5.2.3.2.40. XCP_OK

Purpose	Xcp EB specific API return value when function was successful.
Value	0U

5.2.3.2.41. XCP_OVERLOAD

Purpose	Xcp EB specific API return value when event is already pending (OVERLOAD situation).

Value	4U

5.2.3.2.42. XCP_PROGRAMMING_SEGMENT_NOT_ACTIVE

Purpose	Error Code.
Value	0x86U
Description	The error is triggered by Xcp_SetAbortProgrammingSession() because the programming segment is not active anymore. Therefore the programming sequence shall be aborted and the user can retry again programming the whole sequence.

5.2.3.2.43. XCP_RESP_CTO_QUEUE_FULL

Purpose	Error Code.
Value	0x84U
Description	The error is triggered by Xcp_SendErrorCmdSyntaxPacket() if the response CTO queue, used to enqueue the response packet, is full. Therefore the negative response XCP_ERR_CMD_SYNTAX packet is not enqueued and this matter should be signalled.

5.2.3.2.44. XCP_SID_APPL_COMMAND

Purpose	XCP API service ID.
Value	0x85U
Description	Definition of service ID for Xcp_Appl<CommandName>() API.

5.2.3.2.45. XCP_SID_CRC_MAIN_FUNCTION

Purpose	XCP API service ID.
Value	255U
Description	Definition of service ID for Xcp_CrcMainFunction() API.

5.2.3.2.46. XCP_SID_ENQUEUECTOEVENT

Purpose	XCP API service ID.

Value	0x84U
Description	Definition of service ID for Xcp_PrepareStoredDaqListsResume() API.

5.2.3.2.47. XCP_SID_EVENT_MAIN_FUNCTION

Purpose	XCP API service ID.
Value	255U
Description	Definition of service ID for Xcp_Event_MainFunction() API.

5.2.3.2.48. XCP_SID_GET_VERSION_INFO

Purpose	AUTOSAR API service ID.
Value	0x01U
Description	Definition of service ID for Xcp_GetVersionInfo().

5.2.3.2.49. XCP_SID_IF_RX_INDICATION

Purpose	AUTOSAR API service ID.
Value	0x03U
Description	Definition of the general service ID for Xcp_<If>RxIndication() functions.

5.2.3.2.50. XCP_SID_IF_TRIGGER_TRANSMIT

Purpose	AUTOSAR API service ID.
Value	0x41U
Description	Definition of service ID for Xcp_<If>TriggerTransmit().

5.2.3.2.51. XCP_SID_IF_TX_CONFIRMATION

Purpose	AUTOSAR API service ID.

Value	0x02U
Description	Definition of the general service ID for Xcp_<If>TxConfirmation() functions.

5.2.3.2.52. XCP_SID_INIT

Purpose	AUTOSAR API service ID.
Value	0x00U
Description	Definition of service ID for Xcp_Init().

5.2.3.2.53. XCP_SID_INSERT_ODT_TO_STIM_BUFFER

Purpose	XCP API service ID.
Value	0x87U
Description	Definition of service ID for Xcp_InsertOdtToStimBuffer() API.

5.2.3.2.54. XCP_SID_MAIN_FUNCTION

Purpose	AUTOSAR API service ID.
Value	0x04U
Description	Definition of service ID for Xcp_MainFunction().

5.2.3.2.55. XCP_SID_RX_MAIN_FUNCTION

Purpose	XCP API service ID.
Value	255U
Description	Definition of service ID for Xcp_RxMainFunction() API.

5.2.3.2.56. XCP_SID_SEND_CMD_SYNTAX_PACKET

Purpose	XCP API service ID.
Value	0x83U

Description	Definition of service ID for Xcp_SendErrorCmdSyntaxPacket().

5.2.3.2.57. XCP_SID_SET_ABORT_PROGRAMMING_SESSION

Purpose	XCP API service ID.
Value	0x86U
Description	Definition of service ID for Xcp_SetAbortProgrammingSession() API.

5.2.3.2.58. XCP_SID_SET_EVENT

Purpose	XCP API service ID.
Value	0x80U
Description	Definition of service ID for Xcp_SetEvent() API.

5.2.3.2.59. XCP_SID_SET_TRANSMISSION_MODE

Purpose	AUTOSAR API service ID.
Value	0x05U
Description	Definition of service ID for Xcp_SetTransmissionMode().

5.2.3.2.60. XCP_SID_SOAD_SO_CON_MODE_CHG

Purpose	AUTOSAR API service ID.
Value	0x81U
Description	Definition of service ID for Xcp_SoAdSoConModeChg().

5.2.3.2.61. XCP_SID_STATE_CONTROL

Purpose	AUTOSAR API service ID.
Value	0x83U
Description	Definition of service ID for Xcp_Control().

5.2.3.2.62. XCP_SID_TX_MAIN_FUNCTION

Purpose	XCP API service ID.
Value	255U
Description	Definition of service ID for Xcp_TxMainFunction() API.

5.2.3.2.63. XCP_SID_UPDATE_AVAILABLE_TX_BUFFERS

Purpose	XCP API service ID.
Value	0x88U
Description	Definition of service ID for Xcp_UpdateAvailableTxBuffers() API.

5.2.3.2.64. XCP_STATE_ACTIVE

Purpose	Defines XCP active state where the code is generated as usual.
Value	0xAAU

5.2.3.2.65. XCP_STATE_INACTIVE

Purpose	Defines XCP inactive state where the code is generated but the suppressed functions perform no actions.
Value	0x55U

5.2.3.2.66. XCP_STIM_TIMEOUT

Purpose	Xcp EB specific API return value when not all stimulation packages are received (STIM_TIMEOUT situation).
Value	6U

5.2.3.2.67. XCP_SW_MAJOR_VERSION

Purpose	AUTOSAR module major version.

Value	2U

5.2.3.2.68. XCP_SW_MINOR_VERSION

Purpose	AUTOSAR module minor version.
Value	12U

5.2.3.2.69. XCP_SW_PATCH_VERSION

Purpose	AUTOSAR module patch version.
Value	2U

5.2.3.2.70. XCP_TX_OFF

Purpose	Transmission Disabled.
Value	0U

5.2.3.2.71. XCP_TX_ON

Purpose	Transmission Enabled.
Value	1U

5.2.3.2.72. XCP_USE_MEMORY_PROXY

Purpose	
Value	STD_ON

5.2.3.2.73. XCP_VENDOR_ID

Purpose	AUTOSAR vendor identification: Elektrobit Automotive GmbH.
Value	1U

5.2.3.3. Functions

5.2.3.3.1. Xcp_ApplBuildChecksum

Purpose	Callout function for command BUILD_CHECKSUM.	
Synopsis	`Xcp_ApplReturnType` **`Xcp_ApplBuildChecksum`** `( const void * AddressPtr , uint32 BlockSize , uint32 * ChecksumResult , uint8 * ChecksumType );`	
Sync/Async	Synchronous	
Reentrancy	Non-Reentrant	
Parameters (in)	`AddressPtr`	Starting address for calculating the checksum
	`BlockSize`	Block size in AG
Parameters (out)	`ChecksumResult`	Will return the calculated checksum if ok OR the maximum allowed block size in case of a ERR_OUT_OF_RANGE error for the BlockSize parameter
	`ChecksumType`	Checksum type
Return Value	Function result.	
	`XCP_APPL_OK`	Function successful.
	`XCP_APPL_ERR_OUT_OF_RANGE`	Parameter BlockSize is invalid (too large).
	`XCP_APPL_ERR_ACCESS_DENIED`	Memory access not allowed.
Description	This callout function shall be used to calculate the checksum for a given memory block. Note: if the function returns a different value than those listed in the return value section, the Xcp module will assume that the command was not executed and send the error ERR_CMD_UNKNOWN as response to the master. Implementation of this function shall be provided by the user. The file Xcp_Callouts.c from the templates folder can be used as a template.	

5.2.3.3.2. Xcp_ApplCalPagInit

Purpose	Callout function for Calibration Page and Segment Initialization.

Synopsis	`void` **`Xcp_ApplCalPagInit`** `( void );`
Sync/Async	Synchronous
Reentrancy	Non-Reentrant
Description	This callout function shall do the initialization of all Segments and Pages. Implementation of this function shall be provided by the user. The file Xcp_Callouts.c from the templates folder can be used as a template.

5.2.3.3.3. Xcp_ApplCompareKey

Purpose	Compare Key from the Master.	
Synopsis	`Std_ReturnType` **`Xcp_ApplCompareKey`** `( uint8 ResourceIdentifier , const uint8 * KeyBufferPtr , uint8 KeyLength );`	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different ResourceIdentifier. Non-reentrant for the same ResourceIdentifier	
Parameters (in)	`ResourceIdentifier`	A Resource to be unlocked.
	`KeyBufferPtr`	Pointer to the key to be verified.
	`KeyLength`	Length of the key RAM.
Return Value	Function result	
	`E_OK`	Function successful
	`E_NOT_OK`	Function not successful
Description	This function will be called by the XCP Slave when the key from master needs to be compared. Implementation of this function shall be provided by the user. The file Xcp_Callouts.c from the templates folder can be used as a reference. The function shall implement the algorithm being used by the master to generate the Key.	

5.2.3.3.4. Xcp_ApplCopyCalPage

Purpose	Callout function for command COPY_CAL_PAGE.
Synopsis	`Xcp_ApplReturnType` **`Xcp_ApplCopyCalPage`** `( uint8 SrcSegment , uint8 SrcPage , uint8 DestSegment , uint8 DestPage );`
Sync/Async	Synchronous

Reentrancy	Non-Reentrant	
Parameters (in)	`SrcSegment`	Logical data segment number source
	`SrcPage`	Logical data page number source
	`DestSegment`	Logical data segment number destination
	`DestPage`	Logical data page number destination
Return Value	Function result.	
	`XCP_APPL_OK`	Function successful.
	`XCP_APPL_ERR_WRITE_PROTECTED`	Destination is write protected because it is a Flash segment.
	`XCP_APPL_ERR_PAGE_NOT_VALID`	Parameter SrcPage or DestPage is invalid.
	`XCP_APPL_ERR_SEGMENT_NOT_VALID`	Parameter SrcSegment or DestSegment is invalid.
Description	This callout function shall copy the given source page to the given target page. Note: if the function returns a different value than those listed in the return value section, the Xcp module will assume that the command was not executed and send the error ERR_CMD_UNKNOWN as response to the master. Implementation of this callout function shall be provided by the user. The file Xcp_Callouts.c from the templates folder can be used as a template.	

5.2.3.3.5. Xcp_ApplGetAddress

Purpose	Callout function for address translation.	
Synopsis	`void * Xcp_ApplGetAddress ( uint8 AddressExtension , void * AddressPtr );`	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	`AddressExtension`	8 bit XCP address extension. It is always called with value 0 as the address extension feature is not supported.
	`AddressPtr`	Pointer to the XCP address
Return Value	Pointer to the physical address to access.	
Description	This callout function shall translate the given XCP address and the address extension into the physical address to access.	

	Implementation of this function shall be provided by the user. The file Xcp_Callouts.c from the templates folder can be used as a template.

5.2.3.3.6. Xcp_ApplGetCalPage

Purpose	Callout function for command GET_CAL_PAGE.	
Synopsis	`Xcp_ApplReturnType` **`Xcp_ApplGetCalPage`** `( uint8 Segment , uint8 Mode , uint8 * PagePtr );`	
Sync/Async	Synchronous	
Reentrancy	Non-Reentrant	
Parameters (in)	`Segment`	Logical data segment number
	`Mode`	Access mode
Parameters (out)	`PagePtr`	Logical data page number to be returned to the master.
Return Value	Function result.	
	`XCP_APPL_OK`	Function successful.
	`XCP_APPL_ERR_MODE_NOT_VALID`	Parameter Mode is invalid.
	`XCP_APPL_ERR_SEGMENT_NOT_VALID`	Parameter Segment is invalid.
	`XCP_APPL_ERR_PAGE_NOT_VALID`	Parameter PagePtr is invalid.
Description	This callout function shall get the logical data page number belonging to the given logical segment (Segment) and to the Mode parameter, and provide it in the parameter PagePtr. Note: if the function returns a different value than those listed in the return value section, the Xcp module will assume that the command was not executed and send the error ERR_CMD_UNKNOWN as response to the master. Implementation of this function shall be provided by the user. The file Xcp_Callouts.c from the templates folder can be used as a template.	

5.2.3.3.7. Xcp_ApplGetPagProcessorInfo

Purpose	Callout function for command GET_PAG_PROCESSOR_INFO.
Synopsis	`Xcp_ApplReturnType` **`Xcp_ApplGetPagProcessorInfo`** `( uint8 * MaxSegmentPtr , uint8 * PagPropertiesPtr );`

Sync/Async	Synchronous	
Reentrancy	Non-Reentrant	
Parameters (out)	`MaxSegmentPtr`	Total number of available segments to be returned to the master
	`PagPropertiesPtr`	General properties for paging (freeze mode) to be returned to the master.
Return Value	Function result.	
	`XCP_APPL_OK`	Function successful.
Description	This callout function provides the paging processor basic information: total number of available segments and if freezing mode is supported. Note: if the function returns a different value than those listed in the return value section, the Xcp module will assume that the command was not executed and send the error ERR_CMD_UNKNOWN as response to the master. Implementation of this function shall be provided by the user. The file Xcp_Callouts.c from the templates folder can be used as a template.	

5.2.3.3.8. Xcp_ApplGetPageInfo

Purpose	Callout function for command GET_PAGE_INFO.	
Synopsis	`Xcp_ApplReturnType` **`Xcp_ApplGetPageInfo`** `( uint8 Segment , uint8 Page , uint8 * PagePropertiesPtr , uint8 * InitSegmentPtr );`	
Sync/Async	Synchronous	
Reentrancy	Non-Reentrant	
Parameters (in)	`Segment`	Logical data segment number
	`Page`	Logical data page number
Parameters (out)	`PagePropertiesPtr`	Page properties to be returned to the master
	`InitSegmentPtr`	Segment that initializes this Page
Return Value	Function result.	
	`XCP_APPL_OK`	Function successful.
	`XCP_APPL_ERR_SEGMENT_NOT_VALID`	Parameter Segment is invalid.
	`XCP_APPL_ERR_PAGE_NOT_VALID`	Parameter Page is invalid.
	`XCP_APPL_ERR_OUT_OF_RANGE`	Parameter Segment is invalid.

Description	This callout function shall provide the properties and the INIT_SEGMENT of the Page and its associated Segment. This information shall be written to the following addresses: ► PagePropertiesPtr and ► InitSegmentPtr parameter. Note: if the function returns a different value than those listed in the return value section, the Xcp module will assume that the command was not executed and send the error ERR_CMD_UNKNOWN as response to the master. Implementation of this function shall be provided by the user. The file Xcp_Callouts.c from the templates folder can be used as a template.

5.2.3.3.9. Xcp_ApplGetPgmProcessorInfo

Purpose	Callout function for command GET_PGM_PROCESSOR_INFO.	
Synopsis	`Xcp_ApplReturnType` **`Xcp_ApplGetPgmProcessorInfo`** `( uint8 * MaxSectorPtr , uint8 * PgmPropertiesPtr );`	
Sync/Async	Synchronous	
Reentrancy	Non-Reentrant	
Parameters (out)	`MaxSectorPtr`	Total number of available sectors to be returned to the master
	`PgmPropertiesPtr`	General properties for programming to be returned to the master. The programming mode bits will be masked to only contain those modes which are supported by the Xcp (only ABSOLUTE_MODE is supported by the Xcp).
Return Value	Function result.	
	`XCP_APPL_OK`	Function successful.
Description	This callout function provides the programming processor basic information: ► total number of available sectors ► and programming general properties: ► the available mode[s]; ► data compression properties; ► data encryption properties;	

	► and message sequence processing properties. Note: if the function returns a different value than those listed in the return value section, the Xcp module will assume that the command was not executed and send the error ERR_CMD_UNKNOWN as response to the master. Implementation of this function shall be provided by the user. The file Xcp_Callouts.c from the templates folder can be used as a template.

5.2.3.3.10. Xcp_ApplGetSectorInfo

Purpose	Callout function for command GET_SECTOR_INFO.	
Synopsis	`Xcp_ApplReturnType` **`Xcp_ApplGetSectorInfo`** `( uint8 Mode , uint8 Sector , uint8 * MtaPtr , uint8 * CommandResponsePtr );`	
Parameters (in)	`Mode`	Type of information requested for the given Sector number (start address, length [BYTE] or name length).
	`Sector`	The Sector number for which information is requested.
Parameters (out)	`MtaPtr`	Optional parameter used to get the MTA of the SECTOR name. MtaPtr parameter is the address of a 4-byte buffer containing the location address from which the Master may upload the SECTOR name as ASCII text, using one or more UPLOAD commands. Note: This parameter is mandatory in case Mode equals 2.
	`CommandResponsePtr`	Command response buffer address: ► a 7-byte buffer, in case one of the following sector information is requested: the start address or length [BYTE], or ► a 1-byte buffer, in case the name length sector information is requested;
Return Value	Function result.	
	`XCP_APPL_OK`	Function successful.
	`XCP_APPL_ERR_MODE_NOT_VALID`	Parameter Mode is invalid.

	XCP_APPL_ERR_SEGMENT_NOT_VALID	Parameter Sector is invalid.
Description	This callout function shall provide the sector information that was requested. That information shall be written to the address, to which CommandResponsePtr points to. Note: if the function returns a different value than those listed in the return value section, the Xcp module will assume that the command was not executed and send the error ERR_CMD_UNKNOWN as response to the master. Implementation of this function shall be provided by the user. The file Xcp_Callouts.c from the templates folder can be used as a template. Implementation hints: The following ASAM requirement has to fulfilled by this callout: "The clear and programming sequence number of a flash sector shall be unique and continuous." where, the callout function stores its response data depending on Mode input parameters. ► Depending on the Mode input parameter, this callout function shall provide the following data in the command response buffer: ► Mode 0: ► Byte 0: Clear Sequence Number; ► Byte 1: Program Sequence Number; ► Byte 2: Programming method. ► Bytes 3-6: Start address for this Sector (DWORD). ► Mode 1: ► Byte 0: Clear Sequence Number; ► Byte 1: Program Sequence Number; ► Byte 2: Programming method. ► Bytes 3-6: Length of this Sector [Byte] (DWORD). ► Mode 2: ► Byte 0: SECTOR_NAME_LENGTH in bytes or 0 if not available;	

5.2.3.3.11. Xcp_ApplGetSeed

Purpose	Generate Seed for a resource.
Synopsis	`Std_ReturnType` **`Xcp_ApplGetSeed`** `( uint8 ResourceIdentifier , uint8 * SeedBufferPtr , uint8 * SeedBufferLengthPtr );`

Sync/Async	Synchronous	
Reentrancy	Reentrant for different ResourceIdentifier. Non-reentrant for the same ResourceIdentifier	
Parameters (in)	`ResourceIdentifier`	A Resource to be unlocked.
Parameters (in,out)	`SeedBufferPtr`	Pointer to the RAM area where the seed needs to be stored.
	`SeedBufferLengthPtr`	Pointer to the RAM area where the seed length needs to be stored.
Return Value	Function result	
	`E_OK`	Function successful
	`E_NOT_OK`	Function not successful
Description	This function is called by the XCP slave to generate seed for a requested resource. Implementation of this function shall be provided by the user. The file Xcp_Callouts.c from the templates folder can be used as a reference. The function shall implement the algorithm being used by the master to generate the Seed. When requesting for a resource the user shall make sure that the resource is supported and allowed. Any attempt to unlock a not supporter resource shall return E_NOT_OK.	

5.2.3.3.12. Xcp_ApplGetSegmentInfo

Purpose	Callout function for command GET_SEGMENT_INFO.	
Synopsis	`Xcp_ApplReturnType` **`Xcp_ApplGetSegmentInfo`** `( uint8 Mode , uint8 Segment , uint8 SegmentInfo , uint8 MappingIndex , uint8 * CommandResponsePtr );`	
Sync/Async	Synchronous	
Reentrancy	Non-Reentrant	
Parameters (in)	`Mode`	Type of information requested (basic, standard or address mapping info)
	`Segment`	Logical data segment number
	`SegmentInfo`	Address range information depending on the Mode input parameter
	`MappingIndex`	Identifier for address mapping range that MAPPING_INFO belongs to
Parameters (out)	`CommandResponsePtr`	Command response buffer address. CommandResponsePtr parameter is the address of a 5-byte buffer where the callout

		function stores its response data / parameters depending on Mode, SegmentInfo and MappingIndex input parameters. ► Depending on the Mode input parameter, this callout function shall provide the following data in the command response buffer: ► Mode 0: ► Bytes 0-3: Basic Info (DWord) - Segment Address / Segment Length; ► Byte 4: Reserved. ► Mode 1: ► Byte 0: Number of Pages; ► Byte 1: Address Extension for this segment; ► Byte 2: Number of mapped address ranges for this segment; ► Byte 3: Compression Method; ► Byte 4: Encryption Method. ► Mode 2: ► Bytes 0-3: Mapping Info (DWord) - Source Address / Destination Address / Length; ► Byte 4: Reserved.
Return Value	Function result.	
	`XCP_APPL_OK`	Function successful.
	`XCP_APPL_ERR_MODE_NOT_VALID`	Parameter Mode is invalid.
	`XCP_APPL_ERR_SEGMENT_NOT_VALID`	Parameter Segment is invalid.
	`XCP_APPL_ERR_OUT_OF_RANGE`	Parameter Segment is invalid or parameter SegmentInfo is invalid for that mode (any of these two values shall be returned XCP_APPL_ERR_SEGMENT_NOT_-

		VALID or XCP_APPL_ERR_OUT_OF_-RANGE. The preferred value is XCP_AP-PL_ERR_OUT_OF_RANGE because this is specified by ASAM).
Description	This callout function shall provide the segment information that was requested. That information shall be written to the address, to which CommandResponsePtr points to. Note: if the function returns a different value than those listed in the return value section, the Xcp module will assume that the command was not executed and send the error ERR_CMD_UNKNOWN as response to the master. Implementation of this function shall be provided by the user. The file Xcp_Callouts.c from the templates folder can be used as a template.	

5.2.3.3.13. Xcp_ApplGetSegmentMode

Purpose	Callout function for command GET_SEGMENT_MODE.	
Synopsis	`Xcp_ApplReturnType` **`Xcp_ApplGetSegmentMode`** `( uint8 Segment , uint8 * ModePtr );`	
Sync/Async	Synchronous	
Reentrancy	Non-Reentrant	
Parameters (in)	`Segment`	Logical data segment number
Parameters (out)	`ModePtr`	(FREEZE Mode Disable/Enable) to be returned to the master. The Mode parameter (ModePtr) is a bit mask: ► Bit 7 - 1: Don't care; ► Bit 0 - FREEZE
Return Value	Function result.	
	`XCP_APPL_OK`	Function successful.
	`XCP_APPL_ERR_OUT_OF_RANGE`	The length of the Segment parameter is invalid.
	`XCP_APPL_ERR_SEGMENT_NOT_VALID`	Parameter Segment is invalid.
Description	This callout function shall provide information regarding that the Segment was or was not set for Freeze mode. Note: if the function returns a different value than those listed in the return value section, the Xcp module will assume that the command was not executed and send the error ERR_CMD_UNKNOWN as response to the master.	

	Implementation of this function shall be provided by the user. The file Xcp_Callouts.c from the templates folder can be used as a template.

5.2.3.3.14. Xcp_ApplGetTimestamp

Purpose	Function prototype providing the timestamp.
Synopsis	`Xcp_TimestampType` **`Xcp_ApplGetTimestamp`** `( void );`
Sync/Async	Synchronous
Reentrancy	Non-Reentrant
Return Value	The Timestamp value
Description	This callout function shall provide the user specific timestamp to the Xcp, if that feature is enabled (i.e. the configuration parameter XcpUserTimestampSupported is set to true). Implementation of this function shall be provided by the user. The file Xcp_Callouts.c from the templates folder can be used as a template.

5.2.3.3.15. Xcp_ApplIsMemoryAreaAccessible

Purpose	Callout function to check if access to the requested memory area is allowed.	
Synopsis	`boolean` **`Xcp_ApplIsMemoryAreaAccessible`** `( const uint8 * MemoryAreaAddress , uint32 MemoryAreaLength , uint8 MemoryAreaType , uint8 MemoryAreaScope );`	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	`MemoryAreaAddress`	Start address of the memory area that needs to be validated
	`MemoryAreaLength`	Length of the memory area that needs to be validated
	`MemoryAreaType`	Type (read/write) of the memory area that needs to be validated
	`MemoryAreaScope`	Scope (commands/DAQ) of the memory area that needs to be validated
Return Value		
	`TRUE`	Requested memory area access is allowed

	FALSE	Requested memory area access is denied
Description	Implementation of this function shall be provided by the user. The file Xcp_Callouts.c from the templates folder can be used as a template.	

5.2.3.3.16. Xcp_ApplProgram

Purpose	Callout function for the command PROGRAM.	
Synopsis	`Xcp_ApplReturnType` **`Xcp_ApplProgram`** `( void * AddressPtr , const uint8 * DataPtr , uint16 DataLength );`	
Parameters (in)	AddressPtr	Pointer to the address of the flash block to be programmed.
	DataPtr	Pointer to a data buffer to be programmed.
	DataLength	Length of the buffer to be programmed (in bytes).
Return Value	Function result	
	XCP_APPL_OK	Function successful
	XCP_APPL_ERR_OUT_OF_RANGE	Parameter is invalid
	XCP_APPL_ERR_ACCESS_DENIED	Memory access not allowed
	XCP_APPL_ERR_MEMORY_OVERFLOW	Memory overflow during execution
Description	This function is called by the Xcp in the following cases, depending on the communication mode: ► normal mode: with each execution of PROGRAM or PROGRAM_MAX commands ► block mode: at the end of an entire block mode sequence (i.e. the last PROGRAM_NEXT) and each time a PROGRAM with number of bytes to be programmed = 0 is called. This function should do the actual programming. When this callout is called with DataLength = 0, it means the end of the programming for the current segment.	

5.2.3.3.17. Xcp_ApplProgramClear

Purpose	Callout function for the command PROGRAM_CLEAR.

Synopsis	`Xcp_ApplReturnType` **`Xcp_ApplProgramClear`** `( void * AddressPtr , uint32 ClearRange );`	
Parameters (in)	`AddressPtr`	Pointer to the address of the flash block to be cleared
	`ClearRange`	Length, in bytes, of the block to be cleared
Return Value	Function result	
	`XCP_APPL_OK`	Function successful
	`XCP_APPL_ERR_OUT_OF_RANGE`	Parameter is invalid
	`XCP_APPL_ERR_ACCESS_DENIED`	Memory access not allowed
Description	This function is called by the Xcp whenever the command PROGRAM_CLEAR is executed. Use this function to clear a part of non-volatile memory prior to reprogramming.	

5.2.3.3.18. Xcp_ApplProgramReset

Purpose	Callout function for the command PROGRAM_RESET.
Synopsis	`void` **`Xcp_ApplProgramReset`** `( void );`
Description	This function is called by the Xcp whenever the command PROGRAM_RESET is executed.

5.2.3.3.19. Xcp_ApplProgramStart

Purpose	Callout function for the command PROGRAM_START.	
Synopsis	`Xcp_ApplReturnType` **`Xcp_ApplProgramStart`** `( uint16 * ErrorCodePtr );`	
Parameters (out)	`ErrorCodePtr`	Pointer to write the slave specific error code to. Value is used only to fill the error message ERR_GENERIC. If XCP_APPL_ERR_GENERIC is not thrown by the callout function this parameter is ignored.
Return Value	Function result	
	`XCP_APPL_OK`	Function successful
	`XCP_APPL_ERR_GENERIC`	The slave device is not in a state which permits programming
Description	This function is called by the Xcp whenever the command PROGRAM_START is executed. Use this callout to setup the programming (memory programming may have	

	implementation specific preconditions (slave device in a secure physical state, additional code downloaded, ...).

5.2.3.3.20. Xcp_ApplReadDataFromRAM

Purpose	Function prototype providing the feature read data from RAM memory.	
Synopsis	`Xcp_ApplReturnType` **`Xcp_ApplReadDataFromRAM`** `( void * AddressPtr , const uint8 * DataPtr , uint8 DataLength );`	
Sync/Async	Synchronous	
Reentrancy	Non-Reentrant	
Parameters (in)	`AddressPtr`	The address from where data shall be read from RAM.
	`DataPtr`	Pointer to a data/buffer to which the data shall be read.
	`DataLength`	Number of bytes to be read.
Return Value	Function result	
	`XCP_APPL_OK`	Function successful
	`XCP_APPL_ERR_OUT_OF_RANGE`	Parameter Segment is invalid.
	`XCP_APPL_ERR_ACCESS_DENIED`	The memory location is not accessible.
	`XCP_ERR_CMD_UNKNOWN`	Unknown command or not implemented optional command.
Description	This callout function shall provide the user specific reading data from RAM, if that feature is enabled (i.e. the configuration parameter XcpBlockWriteReadMemoryRAMMechanism is set to true). Implementation of this function shall be provided by the user. This callout is not protected for data access concurrency issues. Therefore, the user shall implement the necessary countermeasures to protect copying data access against possible concurrency issues, if applicable. The file Xcp_Callouts.c from the templates folder can be used as a template.	

5.2.3.3.21. Xcp_ApplSetCalPage

Purpose	Callout function for command SET_CAL_PAGE.

Synopsis	`Xcp_ApplReturnType` **`Xcp_ApplSetCalPage`** `( uint8 Segment , uint8 Page , uint8 Mode );`	
Sync/Async	Synchronous	
Reentrancy	Non-Reentrant	
Parameters (in)	`Segment`	Logical data segment number
	`Page`	Logical data page number
	`Mode`	Mode
Return Value	Function result.	
	`XCP_APPL_OK`	Function successful.
	`XCP_APPL_ERR_PAGE_NOT_VALID`	Parameter Page is invalid.
	`XCP_APPL_ERR_MODE_NOT_VALID`	Parameter Mode is invalid.
	`XCP_APPL_ERR_SEGMENT_NOT_VALID`	Parameter Segment is invalid.
Description	This callout function shall set the Page for the Segment as the active page for the indicated Mode access of that segment. Note: if the function returns a different value than those listed in the return value section, the Xcp module will assume that the command was not executed and send the error ERR_CMD_UNKNOWN as response to the master. Implementation of this function shall be provided by the user. The file Xcp_Callouts.c from the templates folder can be used as a template.	

5.2.3.3.22. Xcp_ApplSetReqStoreCalReq

Purpose	Callout function for command SET_REQUEST - STORE_CAL_REQ mode.	
Synopsis	`Xcp_ApplReturnType` **`Xcp_ApplSetReqStoreCalReq`** `( void );`	
Sync/Async	Asynchronous	
Reentrancy	Non-Reentrant	
Return Value	Function result.	
	`XCP_APPL_OK`	Function successful.
	`XCP_APPL_ERR_OUT_OF_RANGE`	Mode is not supported.
Description	This callout function is a user specific function for the command SET_REQUEST if the flag STORE_CAL_REQ is set. It shall trigger the job of that command, i.e. the storage of the calibration data into the non-volatile memory and it should be done asynchronously not to block the Xcp.	

	Notes: ► when the job is finished, i.e. the data are stored in the non-volatile memory, the callback function Xcp_SetReqStoreCalReqCbk() shall be called (e.g. by this call-out function if it is synchronous or by some other function if it is asynchronous); ► if the function returns a different value than those listed in the return value section, the Xcp module will assume that the command was not executed and send the error ERR_CMD_UNKNOWN as response to the master. Implementation of this function shall be provided by the user. The file Xcp_Callouts.c from the templates folder can be used as a template.

5.2.3.3.23. Xcp_ApplSetSegmentMode

Purpose	Callout function for command SET_SEGMENT_MODE.	
Synopsis	`Xcp_ApplReturnType` **`Xcp_ApplSetSegmentMode`** `( uint8 Segment , uint8 Mode );`	
Sync/Async	Synchronous	
Reentrancy	Non-Reentrant	
Parameters (in)	`Segment`	Logical data segment number
	`Mode`	(FREEZE Mode Disable/Enable) The Mode parameter is a bit mask: ► Bit 7 - 1: Don't care; ► Bit 0 - FREEZE: ► 0 = Disable FREEZE Mode; ► 1 = Enable FREEZE Mode.
Return Value	Function result.	
	`XCP_APPL_OK`	Function successful.
	`XCP_APPL_ERR_OUT_OF_RANGE`	Parameter Segment is invalid.
	`XCP_APPL_ERR_SEGMENT_NOT_VALID`	Parameter Segment is invalid (any of these two values shall be returned XCP_APPL_ERR_SEGMENT_NOT_VALID or XCP_APPL_ERR_OUT_OF_RANGE. The preferred value is XCP_APPL_ERR_OUT_OF_RANGE because this is specified by ASAM).

	XCP_APPL_ERR_MODE_NOT_VALID	Parameter Mode is invalid.
Description	This callout function shall set the freeze mode flag of the segment Segment. Note: if the function returns a different value than those listed in the return value section, the Xcp module will assume that the command was not executed and send the error ERR_CMD_UNKNOWN as response to the master. Implementation of this function shall be provided by the user. The file Xcp_Callouts.c from the templates folder can be used as a template.	

5.2.3.3.24. Xcp_ApplWriteDataToRAM

Purpose	Function prototype providing the feature write data to RAM memory.	
Synopsis	`Xcp_ApplReturnType` **`Xcp_ApplWriteDataToRAM`** `( void * AddressPtr , const uint8 * DataPtr , uint8 DataLength );`	
Sync/Async	Synchronous	
Reentrancy	Non-Reentrant	
Parameters (in)	AddressPtr	The address to which the data shall be written to RAM.
	DataPtr	Pointer to a data/buffer to be written.
	DataLength	Number of bytes to be written.
Return Value	Function result	
	XCP_APPL_OK	Function successful
	XCP_APPL_ERR_OUT_OF_RANGE	Parameter Segment is invalid.
	XCP_ERR_CMD_UNKNOWN	Unknown command or not implemented optional command
Description	This callout function shall provide the user specific writing data to RAM, if that feature is enabled (i.e. the configuration parameter XcpBlockWriteReadMemoryRAMMechanism is set to true). Implementation of this function shall be provided by the user. This callout is not protected for data access concurrency issues. Therefore, the user shall implement the necessary countermeasures to protect copying data access against possible concurrency issues, if applicable. The file Xcp_Callouts.c from the templates folder can be used as a template.	

5.2.3.3.25. Xcp_CanIfRxIndication

Purpose	XCP PDU is received.	
Synopsis	`void` **`Xcp_CanIfRxIndication`** `( PduIdType XcpRxPduId , PduInfoType * XcpRxPduPtr );`	
Service ID	XCP_SID_IF_RX_INDICATION	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different XcpRxPduId. Non-reentrant for the same XcpRxPduId	
Production Errors	▶ XCP_E_PDU_LOST: thrown, if an XCP message is received but it is discarded by XCP because it is detected as incorrect. ▶ XCP_E_PDU_BUFFER_FULL: thrown, if an XCP message is received but it is discarded because the reception buffer is already full. ▶ XCP_E_PDU_BUFFER_LOCKED: thrown, if the callback function was called for a PDU that is not mapped to the active connection. ▶ XCP_E_RESP_CTO_QUEUE_FULL: thrown, if a command response or event package cannot be sent to the master because the internal queue is full. ▶ XCP_E_STIMULATION_DATA_LOST: thrown, if new stimulation data is received before previous data is processed. ▶ XCP_E_ACQUISITION_DATA_LOST: thrown, if the requested acquisition data exceeds the connection MAX_DTO and cannot fit into a package.	
Parameters (in)	`XcpRxPduId`	PDU-ID that has been received
	`XcpRxPduPtr`	Contains the length (SduLength) of the received I-PDU and a pointer to a buffer (SduDataPtr) containing the I-PDU
Description	This function is called by the lower layer CanIf when an AUTOSAR XCP PDU has been received. Precondition: XCP_ON_CAN_ENABLED = STD_ON	

5.2.3.3.26. Xcp_CanIfTxConfirmation

Purpose	XCP PDU transmit confirmation.
Synopsis	`void` **`Xcp_CanIfTxConfirmation`** `( PduIdType XcpTxPduId );`
Service ID	XCP_SID_IF_TX_CONFIRMATION
Sync/Async	Synchronous
Reentrancy	Reentrant for different XcpTxPduId. Non-Reentrant for the same XcpTxPduId

Production Errors	▶ XCP_E_RETRY_FAILED: thrown, if the XCP message transmission (including the configured number of retries) fails. ▶ XCP_E_PDU_BUFFER_LOCKED: thrown, if the callback function was called for a PDU that is not mapped to the active connection.	
Parameters (in)	`XcpTxPduId`	PDU-ID that has been transmitted
Description	This function is called by the lower layer CanIf when an AUTOSAR XCP PDU has been transmitted. Precondition: XCP_ON_CAN_ENABLED = STD_ON	

5.2.3.3.27. Xcp_CddRxIndication

Purpose	XCP PDU is received.	
Synopsis	`void Xcp_CddRxIndication ( PduIdType XcpRxPduId , PduInfoType * XcpRxPduPtr );`	
Service ID	XCP_SID_IF_RX_INDICATION	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different XcpRxPduId. Non-Reentrant for the same XcpRxPduId	
Parameters (in)	`XcpRxPduId`	PDU-ID that has been received.
	`XcpRxPduPtr`	Pointer to SDU(Buffer of received pay-load).
Description	This function is called by the lower layer Complex Device Driver when an AUTOSAR XCP PDU has been received. Precondition: XCP_ON_CDD_ENABLED = STD_ON	

5.2.3.3.28. Xcp_CddTriggerTransmit

Purpose	Call when an AUTOSAR XCP PDU shall be transmitted.
Synopsis	`Std_ReturnType Xcp_CddTriggerTransmit ( PduIdType XcpTxPduId , PduInfoType * PduInfoPtr );`
Service ID	XCP_SID_IF_TRIGGER_TRANSMIT
Sync/Async	Synchronous
Reentrancy	Non-Reentrant

Parameters (in)	`XcpTxPduId`	PDU-ID that has been requested to be transmitted.
Parameters (out)	`PduInfoPtr`	Pointer to PduInfo SDU (pointer to SDU buffer and SDU length).
Return Value		
Description	This function is called by the lower layer Complex Device Driver when an AUTOSAR XCP PDU shall be transmitted. The function Xcp_CddTriggerTransmit is called by the Complex Device Driver for requesting the I-PDU before transmission. Whether the function Xcp_CddTriggerTransmit is called or not is statically configured for each I-PDU in the configuration. Precondition: XCP_ON_CDD_ENABLED = STD_ON	

5.2.3.3.29. Xcp_CddTxConfirmation

Purpose	XCP PDU transmit confirmation.	
Synopsis	`void` **`Xcp_CddTxConfirmation`** `( PduIdType XcpTxPduId );`	
Service ID	XCP_SID_IF_TX_CONFIRMATION	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different XcpTxPduId. Non-Reentrant for the same XcpTxPduId	
Parameters (in)	`XcpTxPduId`	PDU-ID that has been transmitted.
Description	This function is called by the lower layer Complex Device Driver when an AUTOSAR XCP PDU has been transmitted. Precondition: XCP_ON_CDD_ENABLED = STD_ON	

5.2.3.3.30. Xcp_Control

Purpose	Function to perform an XCP disable or enable depending on on the command passed. On disable state the function will terminate the current XCP session and re-set internal data structures and sub processors. The code will be generated no action performed form the suppressed functions. On enable state the Xcp module is reinitialized, the suppressed functionality in disable state is resumed. The function shall be called with the values 0xAA for active, and 0x55 for inactive state.
Synopsis	`void` **`Xcp_Control`** `( uint8 command );`
Service ID	XCP_SID_STATE_CONTROL

Sync/Async	Synchronous	
Reentrancy	Non-Reentrant	
Parameters (in)	`command`	gives the command to enable or disable the Xcp module.
Description	Precondition: The XcpEnableXcpControlApi parameter has to be set to true, thus, XCP_ENABLE_XCP_CONTROL_API is set to STD_ON allowing the Xcp_Control() function to be enabled and called when needed.	

5.2.3.3.31. Xcp_FrIfRxIndication

Purpose	XCP PDU is received.	
Synopsis	`void` **`Xcp_FrIfRxIndication`** `( PduIdType XcpRxPduId , PduInfoType * XcpRxPduPtr );`	
Service ID	XCP_SID_IF_RX_INDICATION	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different XcpRxPduId. Non-reentrant for the same XcpRxPduId	
Production Errors	▶ XCP_E_PDU_LOST: thrown, if an XCP message is received but it is discarded by XCP because it is detected as incorrect. ▶ XCP_E_PDU_BUFFER_FULL: thrown, if an XCP message is received but it is discarded because the reception buffer is already full. ▶ XCP_E_PDU_BUFFER_LOCKED: thrown, if the callback function was called for a PDU that is not mapped to the active connection. ▶ XCP_E_RESP_CTO_QUEUE_FULL: thrown, if a command response or event package cannot be sent to the master because the internal queue is full. ▶ XCP_E_STIMULATION_DATA_LOST: thrown, if new stimulation data is received before previous data is processed. ▶ XCP_E_ACQUISITION_DATA_LOST: thrown, if the requested acquisition data exceeds the connection MAX_DTO and cannot fit into a package.	
Parameters (in)	`XcpRxPduId`	PDU-ID that has been received
	`XcpRxPduPtr`	Contains the length (SduLength) of the received I-PDU and a pointer to a buffer (SduDataPtr) containing the I-PDU
Description	This function is called by the lower layer FrIf when an AUTOSAR XCP PDU has been received. Precondition: XCP_ON_FLEXRAY_ENABLED = STD_ON	

5.2.3.3.32. Xcp_FrIfTriggerTransmit

Purpose	Call when an AUTOSAR XCP PDU shall be transmitted.	
Synopsis	`Std_ReturnType` **`Xcp_FrIfTriggerTransmit`** `( PduIdType XcpTxPduId , PduInfoType * PduInfoPtr );`	
Service ID	XCP_SID_IF_TRIGGER_TRANSMIT	
Sync/Async	Synchronous	
Reentrancy	Non-Reentrant	
Production Errors	▶ XCP_E_PDU_BUFFER_LOCKED: thrown, if the callback function was called for a PDU that is not mapped to the active connection.	
Parameters (in)	`XcpTxPduId`	PDU-ID that has been requested to be transmitted
Parameters (out)	`PduInfoPtr`	Pointer to PduInfo SDU (pointer to SDU buffer and SDU length)
Return Value	Function result	
	`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	This function is called by the lower layer FrIf when an AUTOSAR XCP PDU shall be transmitted. The function Xcp_FrIfTriggerTransmit is called by the Flexray Interface for requesting the I-PDU before transmission. Whether the function Xcp_FrIfTriggerTransmit is called or not is statically configured for each I-PDU in the configuration. Precondition: XCP_ON_FLEXRAY_ENABLED = STD_ON	

5.2.3.3.33. Xcp_FrIfTxConfirmation

Purpose	XCP PDU transmit confirmation.
Synopsis	`void` **`Xcp_FrIfTxConfirmation`** `( PduIdType XcpTxPduId );`
Service ID	XCP_SID_IF_TX_CONFIRMATION
Sync/Async	Synchronous

Reentrancy	Reentrant for different XcpTxPduId. Non-reentrant for the same XcpTxPduId	
Production Errors	▶ XCP_E_RETRY_FAILED: thrown, if the XCP message transmission (including the configured number of retries) fails. ▶ XCP_E_PDU_BUFFER_LOCKED: thrown, if the callback function was called for a PDU that is not mapped to the active connection.	
Parameters (in)	`XcpTxPduId`	PDU-ID that has been transmitted
Description	This function is called by the lower layer FrIf when an AUTOSAR XCP PDU has been transmitted. Precondition: XCP_ON_FLEXRAY_ENABLED = STD_ON	

5.2.3.3.34. Xcp_GetVersionInfo

Purpose	Return the modules version information.	
Synopsis	`void Xcp_GetVersionInfo ( Std_VersionInfoType * VersionInfoPtr );`	
Service ID	XCP_SID_GET_VERSION_INFO	
Sync/Async	Synchronous	
Reentrancy	Non-Reentrant	
Parameters (out)	`VersionInfoPtr`	Pointer to where to store the version information of this module.
Description	This function provides the information to module vendor ID, module ID and software version major.minor.patch Precondition: XCP_VERSION_INFO_API = STD_ON	

5.2.3.3.35. Xcp_Init

Purpose	Initializes the XCP.	
Synopsis	`void Xcp_Init ( const Xcp_ConfigType * Xcp_ConfigPtr );`	
Service ID	XCP_SID_INIT	
Sync/Async	Synchronous	
Reentrancy	Non-Reentrant	
Parameters (in)	`Xcp_ConfigPtr`	Pointer to a selected configuration structure.

Description	This function initializes interfaces and variables of the AUTOSAR XCP module. Precondition: None.

5.2.3.3.36. Xcp_MainFunction

Purpose	Scheduled function of the XCP module.
Synopsis	`void Xcp_MainFunction ( void );`
Service ID	XCP_SID_MAIN_FUNCTION
Production Errors	▶ XCP_E_RESP_CTO_QUEUE_FULL: thrown, if a command response or event package cannot be sent to the master because the internal queue is full. ▶ XCP_E_PDU_LOST: thrown, if an XCP message is received but it is discarded by XCP because it is detected as incorrect. ▶ XCP_E_PDU_BUFFER_FULL: thrown, if an XCP message is received but it is discarded because the reception buffer is already full. ▶ XCP_E_STIMULATION_DATA_LOST: thrown, if new stimulation data is received before previous data is processed. ▶ XCP_E_ACQUISITION_DATA_LOST: thrown, if the requested acquisition data exceeds the connection MAX_DTO and cannot fit into a package. ▶ XCP_E_ILLEGAL_MEMORY_ACCESS: thrown, if an illegal memory access is performed. ▶ XCP_E_RETRY_FAILED: thrown, if an XCP message transmission (including the configured number of retries) fails.
Description	Scheduled function of the XCP module. Precondition: None.

5.2.3.3.37. Xcp_MainFunction_Event

Purpose	Scheduled functions of the XCP module for processing cyclic event.
Synopsis	`void Xcp_MainFunction_Event ( void );`
Production Errors	▶ XCP_E_RETRY_FAILED: thrown, if the XCP message transmission (including the configured number of retries) fails. ▶ XCP_E_PDU_LOST: thrown, if an XCP message is received but it is discarded by XCP because it is detected as incorrect.

	▶ XCP_E_PDU_BUFFER_FULL: thrown, if an XCP message is received but it is discarded because the reception buffer is already full. ▶ XCP_E_RESP_CTO_QUEUE_FULL: thrown, if a command response or event package cannot be sent to the master because the internal queue is full. ▶ XCP_E_STIMULATION_DATA_LOST: thrown, if new stimulation data is received before previous data is processed. ▶ XCP_E_ACQUISITION_DATA_LOST: thrown, if the requested acquisition data exceeds the connection MAX_DTO and cannot fit into a package. ▶ XCP_E_ILLEGAL_MEMORY_ACCESS: thrown, if an illegal memory access is performed.
Description	This function is configurable and it can be used to process independent events on independent (scheduled) main processing function. Precondition: Event main functions enabled.

5.2.3.3.38. Xcp_MemoryProxyResponseHandler

Purpose	This API is used to report that a MemoryProxy instance has finished its task.	
Synopsis	`void` **`Xcp_MemoryProxyResponseHandler`** `( uint16 * EventId );`	
Sync/Async	Asynchronous	
Reentrancy	Non-Reentrant	
Parameters (in)	`EventId`	The Id of the Event for which the MemoryProxy was invoked.

5.2.3.3.39. Xcp_NvmStoreDaqSingleCbk

Purpose	Callback for a single NvM block request.	
Synopsis	`Std_ReturnType` **`Xcp_NvmStoreDaqSingleCbk`** `( uint8 ServiceId , NvM_RequestResultType JobResult );`	
Sync/Async	Synchronous	
Reentrancy	Non-Reentrant	
Production Errors	▶ XCP_E_RESP_CTO_QUEUE_FULL: thrown, if a command response or event package cannot be sent to the master because the internal queue is full.	
Parameters (in)	`ServiceId`	Unique Service ID of NVRAM manager service

	`JobResult`	Covers the job result of the previous processed single block job
Return Value	Result of the operation	
	`E_OK`	Callback function has been processed successfully
	`E_NOT_OK`	Callback function has not been processed successfully
Description	This callback function shall be called by NvM upon the end of a single block request. The XCP slave will initially call NvM to store the DAQ lists configuration. Upon the storage end this callback will be requested and XCP will send a event to the master - EV_STORE_DAQ or EV_CLEAR_DAQ.	

5.2.3.3.40. Xcp_ReadRamBlockFromNvMNativeCbk

Purpose	Callback for reading the associated NvM block.	
Synopsis	`Std_ReturnType` **`Xcp_ReadRamBlockFromNvMNativeCbk`** `( const void * NvMBuffer );`	
Return Value		
Description	This callback function corresponds to configuration parameter NvMReadRamBlock-FromNvCallback and is triggered by NvM.	

5.2.3.3.41. Xcp_RxMainFunction

Purpose	Scheduled function of the XCP module for processing the reception.
Synopsis	`void` **`Xcp_RxMainFunction`** `( void );`
Production Errors	▶ XCP_E_PDU_LOST: thrown, if an XCP message is received but it is discarded by XCP because it is detected as incorrect. ▶ XCP_E_PDU_BUFFER_FULL: thrown, if an XCP message is received but it is discarded because the reception buffer is already full. ▶ XCP_E_RESP_CTO_QUEUE_FULL: thrown, if a command response or event package cannot be sent to the master because the internal queue is full. ▶ XCP_E_STIMULATION_DATA_LOST: thrown, if new stimulation data is received before previous data is processed.
Description	This function is configurable and it can be used to process the reception of all messages on an independent (scheduled) main processing function.

	Precondition: Rx main function enabled.

5.2.3.3.42. Xcp_SetEvent

Purpose	Function to set a DAQ list sampling event.	
Synopsis	`Xcp_ReturnType` **`Xcp_SetEvent`** `( uint16 EventId );`	
Service ID	XCP_SID_SET_EVENT	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	`EventId`	XCP event channel ID.
Return Value	Result of the operation.	
	`XCP_OK`	Setting the event was successful.
	`XCP_NOT_OK`	Setting the event failed (a development error happened).
	`XCP_NOT_INITIALIZED`	Setting the event failed (module is not initialized).
	`XCP_NOT_CONNECTED`	Setting the event failed (slave is in the state DISCONNECTED).
	`XCP_OVERLOAD`	Setting the event failed (event is already set).
	`XCP_NO_ACTIVE_LIST`	Setting the event failed (no active list is assigned to that event).
Description	This function sets an event for the given event channel to trigger the sampling of the associated DAQ list(s). This event will then be handled by one of the following Xcp_MainFunction() calls. Precondition: The event channel must be a non-cyclic event channel, i.e. the parameter XcpEventChannelTimeCycle is set to 0 for that event channel.	

5.2.3.3.43. Xcp_SetReqStoreCalReqCbk

Purpose	Callback for STORE_CAL_REQ in SET_REQUEST command.
Synopsis	`Std_ReturnType` **`Xcp_SetReqStoreCalReqCbk`** `( uint8 ServiceId , NvM_RequestResultType JobResult );`

Sync/Async	Synchronous	
Reentrancy	Non-Reentrant	
Production Errors	► XCP_E_RESP_CTO_QUEUE_FULL: thrown, if command response or event package cannot be sent to the master because internal queue is full.	
Parameters (in)	`ServiceId`	- Unique Service ID of NVRAM manager service
	`JobResult`	- Covers the job result of the previous processed single block job
Return Value	Result of the operation.	
	`E_OK`	- Callback function has been processed successfully
Description	This callback function shall be called by the application after issuing the SET_REQUEST command (STORE_CAL_REQ bit set) and after the NvM writing is completed (successful or not). Scenario: SET_REQUEST command (STORE_CAL_REQ bit is set) is received by the XCP module. The XCP module calls the application callout function Xcp_ApplSetReqStoreCalReq() that stores the calibration data to NV memory. After the data is stored to NV memory, the application must call the Xcp_SetReqStoreCalReqCbk() function. This callback function resets the STORE_CAL_REQ bit of the Current Session Status parameter in GET_STATUS only if it is called with the ServiceId input parameter as the ID for NvM_WriteBlock() API, otherwise it does nothing. If the callback function is called with JobResult input parameter NVM_REQ_OK, it also sends the EV_STORE_CAL event, otherwise no event is sent.	

5.2.3.3.44. Xcp_SetTransmissionMode

Purpose	This API is used to turn on and off of the TX capabilities of used communication bus channel in XCP module. 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.
Synopsis	`void` **`Xcp_SetTransmissionMode`** `( NetworkHandleType Channel , Xcp_TransmissionModeType Mode );`
Service ID	XCP_SID_SET_TRANSMISSION_MODE
Sync/Async	Synchronous
Reentrancy	Non-Reentrant

Parameters (in)	`Channel`	The Network channel for the used bus communication
	`Mode`	Enabled or disabled Transmission mode Parameters
Description	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.	

5.2.3.3.45. Xcp_SoAdIfRxIndication

Purpose	XCP PDU is received.	
Synopsis	`void` **`Xcp_SoAdIfRxIndication`** `( PduIdType XcpRxPduId , PduInfoType * XcpRxPduPtr );`	
Service ID	XCP_SID_IF_RX_INDICATION	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different XcpRxPduId. Non-reentrant for the same XcpRxPduId.	
Production Errors	▶ XCP_E_PDU_LOST: thrown, if an XCP message is received but it is discarded by XCP because it is detected as incorrect. ▶ XCP_E_PDU_BUFFER_FULL: thrown, if an XCP message is received but it is discarded because the reception buffer is already full. ▶ XCP_E_PDU_BUFFER_LOCKED: thrown, if the callback function was called for a PDU that is not mapped to the active connection. ▶ XCP_E_RESP_CTO_QUEUE_FULL: thrown, if a command response or event package cannot be sent to the master because the internal queue is full. ▶ XCP_E_STIMULATION_DATA_LOST: thrown, if new stimulation data is received before previous data is processed. ▶ XCP_E_ACQUISITION_DATA_LOST: thrown, if the requested acquisition data exceeds the connection MAX_DTO and cannot fit into a package.	
Parameters (in)	`XcpRxPduId`	PDU-ID that has been received
	`XcpRxPduPtr`	Pointer to SDU(Buffer of received payload)
Description	This function is called by the lower layer Socket Adapter when an AUTOSAR XCP PDU has been received. Precondition: XCP_ON_ETHERNET_ENABLED = STD_ON	

5.2.3.3.46. Xcp_SoAdIfTxConfirmation

Purpose	XCP PDU transmit confirmation.	
Synopsis	`void` **`Xcp_SoAdIfTxConfirmation`** `( PduIdType XcpTxPduId );`	
Service ID	XCP_SID_IF_TX_CONFIRMATION	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different XcpTxPduId. Non-reentrant for the same XcpTxPduId.	
Production Errors	► XCP_E_RETRY_FAILED: thrown, if the XCP message transmission (including the configured number of retries) fails. ► XCP_E_PDU_BUFFER_LOCKED: thrown, if the callback function was called for a PDU that is not mapped to the active connection.	
Parameters (in)	`XcpTxPduId`	PDU-ID that has been transmitted
Description	This function is called by the lower layer Socket Adapter when an AUTOSAR XCP PDU has been transmitted. Precondition: XCP_ON_ETHERNET_ENABLED = STD_ON	

5.2.3.3.47. Xcp_SoAdSoConModeChg

Purpose	Socket connection mode change notification.	
Synopsis	`void` **`Xcp_SoAdSoConModeChg`** `( SoAd_SoConIdType SoConId , SoAd_SoConModeType Mode );`	
Service ID	XCP_SID_SOAD_SO_CON_MODE_CHG	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different SoConIds. Non reentrant for the same SoConId	
Production Errors	► XCP_E_RESP_CTO_QUEUE_FULL: thrown, if a command response or event package cannot be sent to the master because the internal queue is full.	
Parameters (in)	`SoConId`	Socket connection ID of socket that has changed
	`Mode`	Socket connection mode (ONLINE, OFFLINE or RECONNECT)
Description	This function is called by the lower layer Socket Adapter when the socket connection mode has changed. If the socket is closed while in XCP connected state, the XCP module will perform an XCP disconnect, which means that all data acquisition will be stopped.	

	This function is available only for the TCP protocol as there is no UDP connection (UDP being a broadcasting protocol) Precondition: XCP_ON_ETHERNET_ENABLED = STD_ON

5.2.3.3.48. Xcp_TxMainFunction

Purpose	Scheduled function of the XCP module for processing the transmission.
Synopsis	`void Xcp_TxMainFunction ( void );`
Production Errors	► XCP_E_RETRY_FAILED: thrown, if the XCP message transmission (including the configured number of retries) fails.
Description	This function is configurable and it can be used to process the transmission of all messages on an independent (scheduled) main processing function. Precondition: Tx main function enabled.

5.2.3.3.49. Xcp_WriteRamBlockToNvMNativeCbk

Purpose	Callback for writing the associated NvM block.	
Synopsis	`Std_ReturnType Xcp_WriteRamBlockToNvMNativeCbk ( void * NvM-Buffer );`	
Return Value		
Description	This callback function corresponds to configuration parameter NvMWriteRamBlock-ToNvCallback and is triggered by NvM.	

5.2.4. Integration notes

5.2.4.1. Exclusive areas

This section describes the exclusive areas used by the `XCP` module.

5.2.4.1.1. SCHM_XCP_EXCLUSIVE_AREA_XCP_INTERNALS

Protected data structures	All shared data that shall be protected from mutual access.

Recommended locking mechanism	This exclusive area must always be protected by a locking mechanism, e.g. `ALL_INTERRUPT_BLOCKING` in the `Rte` module configuration. The options for locking are described in the `EB tresos AutoCore Generic documentation`. Refer to the section `Mapping exclusive areas in the basic software modules` in the `Integration notes` section for details.

5.2.4.2. Production errors

XCP_E_ACQUISITION_DATA_LOST	▶ Xcp_CanIfRxIndication ▶ Xcp_FrIfRxIndication ▶ Xcp_MainFunction ▶ Xcp_MainFunction_Event ▶ Xcp_SoAdIfRxIndication
XCP_E_ILLEGAL_MEMORY_ACCESS	▶ Xcp_MainFunction ▶ Xcp_MainFunction_Event
XCP_E_PDU_BUFFER_FULL	▶ Xcp_CanIfRxIndication ▶ Xcp_FrIfRxIndication ▶ Xcp_MainFunction ▶ Xcp_MainFunction_Event ▶ Xcp_RxMainFunction ▶ Xcp_SoAdIfRxIndication
XCP_E_PDU_BUFFER_LOCKED	▶ Xcp_CanIfRxIndication ▶ Xcp_CanIfTxConfirmation ▶ Xcp_FrIfRxIndication ▶ Xcp_FrIfTriggerTransmit ▶ Xcp_FrIfTxConfirmation ▶ Xcp_SoAdIfRxIndication ▶ Xcp_SoAdIfTxConfirmation
XCP_E_PDU_LOST	▶ Xcp_CanIfRxIndication ▶ Xcp_FrIfRxIndication ▶ Xcp_MainFunction

	▶ Xcp_MainFunction_Event ▶ Xcp_RxMainFunction ▶ Xcp_SoAdIfRxIndication
XCP_E_RESP_CTO_QUEUE_FULL	▶ Xcp_CanIfRxIndication ▶ Xcp_FrIfRxIndication ▶ Xcp_MainFunction ▶ Xcp_MainFunction_Event ▶ Xcp_NvmStoreDaqSingleCbk ▶ Xcp_RxMainFunction ▶ Xcp_SetReqStoreCalReqCbk ▶ Xcp_SoAdIfRxIndication ▶ Xcp_SoAdSoConModeChg
XCP_E_RETRY_FAILED	▶ Xcp_CanIfTxConfirmation ▶ Xcp_FrIfTxConfirmation ▶ Xcp_MainFunction ▶ Xcp_MainFunction_Event ▶ Xcp_SoAdIfTxConfirmation ▶ Xcp_TxMainFunction
XCP_E_STIMULATION_DATA_LOST	▶ Xcp_CanIfRxIndication ▶ Xcp_FrIfRxIndication ▶ Xcp_MainFunction ▶ Xcp_MainFunction_Event ▶ Xcp_RxMainFunction ▶ Xcp_SoAdIfRxIndication

5.2.4.3. Memory mapping

General information about memory mapping is provided in the EB tresos AutoCore Generic documentation. Refer to the section `Memory mapping and compiler abstraction` in the `Integration notes` section for details.

The following table provides the list of sections that may be mapped for this module:

Memory section

CONST_16
CONST_8
CONST_UNSPECIFIED
VAR_16
VAR_8
VAR_CONTROL_API_8
VAR_UNSPECIFIED
VAR_CLEARED_16
VAR_CLEARED_8
VAR_CLEARED_UNSPECIFIED
CODE
APPL_CODE
VAR_SAVED_ZONE_UNSPECIFIED
VAR_SAVED_ZONE_UNSPECIFIED_8
WRAPPER_CODE

5.2.4.4. Integration requirements

WARNING 

Integration requirements list is not exhaustive

The following list of integration requirements helps you to integrate your product. However, this list is not exhaustive. You also require information from the user's guide, release notes, and EB tresos AutoCore known issues to successfully integrate your product.

5.2.4.4.1. lim.Xcp.EB_INTREQ_Xcp_0001

Description	The command GET_ID supports slave device identification only using identification type ASAM-MC2 filename without path and extension. This limitation will prevent the user from identifying the slave by 'ASCII text' or by ASAM-MC2 filename with path and extension, or by ASAM-MC2 URL, or by ASAM-MC2 file to upload.

5.2.4.4.2. lim.Xcp.EB_INTREQ_Xcp_0002

Description	The time stamp values in STIM DTOs transmitted by master are ignored by slave.

Rationale	The stimulation is done with the oldest data sent by Xcp master.

5.2.4.4.3. lim.Xcp.EB_INTREQ_Xcp_0003

Description	DTO transmission rate of the master must be equal or less than the cycle time of the Xcp main function. The DTO transmission rate depends on the consistency configuration of event channels. In case of Event consistency, all the DTOs of an event channel to be processed can be sent. In case of DAQ consistency, all the DTOs of a DAQ list to be processed can be sent. In case of ODT consistency, a single DTO of a DAQ list to be processed can be sent.
Rationale	Stimulation is done in Xcp main function and only one snapshot of the data is kept. Hence data from the master will get ignored in case the data transmission rate of master exceeds the Xcp main function cycle time.

5.2.4.4.4. lim.Xcp.EB_INTREQ_Xcp_0004

Description	The Xcp master has to transfer the address parameter of WRITE_DAQ commands in extended/paged addressing(far) format for the 16-bit slave. That is, the address information transmitted through WRITE_DAQ command must contain additional paging information which is needed in accessing a data placed in extended/paged memory of slave.
Rationale	The variables to be measured/calibrated might be in the extended/paged memory. So the Xcp slave expects the address information with paging information.

5.2.4.4.5. lim.Xcp.EB_INTREQ_Xcp_0005

Description	Sampling of data corresponding to an event channel depends on the consistency of the Event Channel. In DAQ and ODT consistency, the sampling of data will get delayed based on the priority of DAQ list. Delayed sampling never occurs for data corresponding to event channels with EVENT consistency, since all the data corresponding to an Event Channel will be sampled in a single main function call.
Rationale	In case of DAQ consistency, only the data corresponding to a DAQ list of an event channel will be sampled in a single main function call. Hence, multiple main function calls are needed to sample all the data corresponding to the event channel. In case of ODT consistency, only the data corresponding to an ODT of a DAQ list of an event

	channel will be sampled in a single main function call. Hence, multiple main function calls are needed to sample all the data corresponding to the event channel.

5.2.4.4.6. lim.Xcp.EB_INTREQ_Xcp_0006

Description	Fixed event channels can only be configured for predefined DAQ lists.
Rationale	The current implementation only assigns default values for predefined DAQ lists. Setting a fixed event channel for a configurable DAQ list would require a preconfigured event channel assigned.

5.2.4.4.7. lim.Xcp.EB_INTREQ_Xcp_0007

Description	Data acquisition and stimulation commands (see ASAM Part 2 - Protocol Layer Specification, chapter 1.4.4 for list) that can modify a DAQ list and the DISCONNECT command are not accepted during the following pending requests: CLEAR_DAQ_REQ, STORE_DAQ_REQ_RESUME, STORE_DAQ_REQ_NO_RESUME.
Rationale	As the NV memory writing of the DAQ lists is made upon the runtime DAQ lists information and not on some shadow buffer, consistency for these DAQ lists must be ensured. Therefore it is not allowed to modify the DAQ lists and their related information during the NV memory write.

5.2.4.4.8. lim.Xcp.EB_INTREQ_Xcp_0008

Description	If at least one DAQ list is in running state, the following modes for the SET_REQUEST command will be rejected: CLEAR_DAQ_REQ, STORE_DAQ_REQ_-RESUME, STORE_DAQ_REQ_NO_RESUME.
Rationale	As the NV memory writing of the DAQ lists is made upon the runtime DAQ lists information and not on some shadow buffer, consistency for these DAQ lists must be ensured. Therefore it is not allowed to perform the writing since the DAQ lists can be altered during the processing of their associated events.

5.2.4.4.9. lim.Xcp.EB_INTREQ_Xcp_0009

Description	GET_SEED only supports mode 0, i.e. it only supports seeds that fit into a single command transfer object (CTO). Therefore seeds are restricted to a maximum length of MAX_CTO-2 bytes, where MAX_CTO is the maximum length of a CTO as defined in the ASAM XCP specification.

Rationale	UNLOCK supports only keys that fit into a CTO. Therefore keys are restricted to a maximum length of MAX_CTO-2 bytes as well.

5.2.4.4.10. lim.Xcp.EB_INTREQ_Xcp_0010

Description	As the queue size for the event packets is limited, it might be that the queue fills up and further event packets are silently ignored.
Rationale	The queue size is limited and has been chosen taking in consideration the resource consumption of the XCP module.

5.2.4.4.11. lim.Xcp.EB_INTREQ_Xcp_0011

Description	Current implementation does not support the address extension feature.

5.2.4.4.12. lim.Xcp.EB_INTREQ_Xcp_0012

Description	Current implementation supports only Optimisation Type = OM_DEFAULT.

5.2.4.4.13. lim.Xcp.EB_INTREQ_Xcp_0013

Description	The WRITE_DAQ command checks that the size of an ODT entry is less than the maximum between MAX_ODT_ENTRY_SIZE_DAQ and MAX_ODT_ENTRY_SIZE_-STIM. The WRITE_DAQ command does not check if it is less than MAX_ODT_-ENTRY_SIZE_DAQ if the direction is equal to DAQ or less than MAX_ODT_EN-TRY_SIZE_STIM if the direction is equal to STIM.
Rationale	WRITE_DAQ command is, most likely, issued before the direction is set by SET_-DAQ_LIST_MODE command. Even if the direction is set, it can be that the direction will be changed afterwards. Therefore it is not known which direction the ODT entry might have in the end, so just a check occurs against the maximum between those two limits.

5.2.4.4.14. lim.Xcp.EB_INTREQ_Xcp_0014

Description	The Xcp slave will reject further allocation of ODT entries ALLOC_ODT_ENTRY) once their configuration has started (WRITE_DAQ).

Rationale	Run-time performance of Xcp slave is increased as, otherwise, the ODT entries must be re-arranged with a full copy process.

5.2.4.4.15. lim.Xcp.EB_INTREQ_Xcp_0015

Description	The Xcp slave will reject further allocation of ODT entries ALLOC_ODT_ENTRY) once their configuration has started (WRITE_DAQ).
Rationale	PROGRAM_START and PROGRAM_CLEAR are not accepted during an active programming of a segment. PROGRAM_START is not accepted when the programming session has been already started.

5.2.4.4.16. lim.Xcp.EB_INTREQ_Xcp_0016

Description	Xcp will handle the calls to the programming callout functions synchronously. This means that if programming is made asynchronously by the callout function, the Xcp slave will be unaware of that.

5.2.4.4.17. lim.Xcp.EB_INTREQ_Xcp_0017

Description	When the user requests access to resources he shall check that the requested resource is allowed. If the resource is not supported the callout shall return E_NOT_OK.
Rationale	The Xcp slave relies on the callout function to perform requests sanity checks and thus, it is the callout function's job to perform the required sanity checks.

5.2.4.4.18. lim.Xcp.EB_INTREQ_Xcp_0018

Description	From the context of one PDU channel, the Xcp Slave allocates necessary space, to store only one package, until is processed via Xcp_MainFunction(). Hence packages will be lost if the Master transmit rate is higher then the Xcp processing cycle(the cycle of Xcp_MainFunction), on a specific PDU channel.
Rationale	The incoming packages are received via Xcp_<If>RxIndication() API, which is called from transport layer context. It makes sense to use that API only to copy data into the allocated internal buffer for the PDU channel, and to process the incoming data from

	MainFunction context. In this way, the run-time used to process the incoming packages is allocated from the Xcp processing context

5.2.4.4.19. lim.Xcp.EB_INTREQ_Xcp_0019

Description	Although specified by ASAM, error code ERR_RES_TEMP_NOT_A (0x33) is not implemented for command GET_SEED.

5.2.4.4.20. lim.Xcp.EB_INTREQ_Xcp_0020

Description	Although specified by ASAM, error code ERR_RES_TEMP_NOT_A (0x33) is not implemented for commands SET_CAL_PAGE and GET_CAL_PAGE.

5.2.4.4.21. lim.Xcp.EB_INTREQ_Xcp_0021

Description	Although specified by ASAM, error code ERR_RES_TEMP_NOT_A (0x33) is not implemented for command GET_PAGE_INFO.

5.2.4.4.22. lim.Xcp.EB_INTREQ_Xcp_0022

Description	Although specified by ASAM, error code ERR_RES_TEMP_NOT_A (0x33) is not implemented for commands SET_SEGMENT_MODE and GET_SEGMENT_MODE.

5.2.4.4.23. lim.Xcp.EB_INTREQ_Xcp_0023

Description	Although specified by ASAM, error code ERR_RES_TEMP_NOT_A (0x33) is not implemented for command COPY_CAL_PAGE.

5.2.4.4.24. lim.Xcp.EB_INTREQ_Xcp_0024

Description	Although specified by ASAM, error code ERR_RES_TEMP_NOT_A (0x33) is not implemented for command PROGRAM_CLEAR.

5.2.4.4.25. lim.Xcp.EB_INTREQ_Xcp_0025

Description	Although specified by ASAM, error codes ERR_RES_TEMP_NOT_A(0x33), ERR_-MEMORY_OVERFLOW(0x30), ERR_UNKNOWN(0x20) are not implemented for command PROGRAM_MAX.

5.2.4.4.26. lim.Xcp.EB_INTREQ_Xcp_0026

Description	Although specified by ASAM, error code ERR_RES_TEMP_NOT_A (0x33) is not implemented for command GET_SECTOR_INFO.

5.2.4.4.27. lim.Xcp.EB_INTREQ_Xcp_0027

Description	Although specified by ASAM, error code ERR_RES_TEMP_NOT_A (0x33) is not implemented for command GET_COMM_MODE_INFO.

5.2.4.4.28. lim.Xcp.EB_INTREQ_Xcp_0028

Description	Although specified by ASAM, error codes ERR_RES_TEMP_NOT_A(0x33), ERR_-CMD_UNKNOWN(0x20) are not implemented for command GET_ID.

5.2.4.4.29. lim.Xcp.EB_INTREQ_Xcp_0029

Description	Although specified by ASAM, error code ERR_CMD_UNKNOWN(0x20) is not implemented for command SET_REQUEST.

5.2.4.4.30. lim.Xcp.EB_INTREQ_Xcp_0030

Description	Although specified by ASAM, error codes ERR_CMD_UNKNOWN(0x20), ERR_-OUT_OF_RANGE(0x22), ERR_RES_TEMP_NOT_A(0x33) are not implemented for command UNLOCK.

5.2.4.4.31. lim.Xcp.EB_INTREQ_Xcp_0031

Description	Although specified by ASAM, error codes ERR_CMD_UNKNOWN(0x20), ERR_-OUT_OF_RANGE(0x22), ERR_RES_TEMP_NOT_A(0x33) are not implemented

	for command SET_MTA. The following error codes ERR_CMD_UNKNOWN(0x20), ERR_OUT_OF_RANGE(0x22) can be reported but only in case read/write memory call-outs return specific error codes.

5.2.4.4.32. lim.Xcp.EB_INTREQ_Xcp_0032

Description	Although specified by ASAM, error codes ERR_CMD_UNKNOWN(0x20), ERR_ACCESS_LOCKED(0x25), ERR_RES_TEMP_NOT_A(0x33) are not implemented for command UPLOAD. The following error codes ERR_CMD_UNKNOWN(0x20), ERR_OUT_OF_RANGE(0x22), ERR_ACCESS_DENIED(0x24) can be reported but only in case read/write memory call-outs return specific error codes.

5.2.4.4.33. lim.Xcp.EB_INTREQ_Xcp_0033

Description	Although specified by ASAM, error codes ERR_CMD_UNKNOWN(0x20), ERR_ACCESS_LOCKED(0x25), ERR_ACCESS_DENIED(0x24), ERR_RES_TEMP_NOT_A(0x33) are not implemented for command SHORT_UPLOAD. The following error codes ERR_CMD_UNKNOWN(0x20), ERR_ACCESS_DENIED(0x24), ERR_OUT_OF_RANGE(0x22) can be reported but only in case read/write memory call-outs return specific error codes.

5.2.4.4.34. lim.Xcp.EB_INTREQ_Xcp_0034

Description	Although specified by ASAM, error codes ERR_ACCESS_LOCKED(0x25), ERR_RES_TEMP_NOT_A(0x33) are not implemented for command BUILD_CHECKSUM.

5.2.4.4.35. lim.Xcp.EB_INTREQ_Xcp_0035

Description	Although specified by ASAM, error code ERR_RES_TEMP_NOT_A(0x33) is not implemented for command TRANSPORT_LAYER_CMD.

5.2.4.4.36. lim.Xcp.EB_INTREQ_Xcp_0036

Description	Although specified by ASAM, error codes ERR_WRITE_PROTECTED(0x23), ERR_MEMORY_OVERFLOW(0x30), ERR_RES_TEMP_NOT_A(0x33) are not

	implemented for command DOWNLOAD. The following error codes ERR_CMD_-UNKNOWN(0x20), ERR_OUT_OF_RANGE(0x22) can be reported but only in case read/write memory call-outs return specific error codes.

5.2.4.4.37. lim.Xcp.EB_INTREQ_Xcp_0037

Description	Although specified by ASAM, error codes ERR_CMD_UNKNOWN(0x20), ERR_-WRITE_PROTECTED(0x23), ERR_MEMORY_OVERFLOW(0x30), ERR_RES_-TEMP_NOT_A(0x33) are not implemented for command MODIFY_BITS. The following error codes ERR_CMD_UNKNOWN(0x20), ERR_ACCESS_DENIED(0x24), ERR_OUT_OF_RANGE(0x22) can be reported but only in case read/write memory call-outs return specific error codes.

5.2.4.4.38. lim.Xcp.EB_INTREQ_Xcp_0038

Description	Although specified by ASAM, error code ERR_RES_TEMP_NOT_A(0x33) is not implemented for command GET_PAG_PROCESSOR_INFO.

5.2.4.4.39. lim.Xcp.EB_INTREQ_Xcp_0039

Description	Although specified by ASAM, error codes ERR_CMD_UNKNOWN(0x20), ERR_-DAQ_ACTIVE(0x11), ERR_RES_TEMP_NOT_A(0x33) are not implemented for command SET_DAQ_PTR.

5.2.4.4.40. lim.Xcp.EB_INTREQ_Xcp_0040

Description	Although specified by ASAM, error codes ERR_DAQ_CONFIG(0x2A), ERR_RES_-TEMP_NOT_A(0x33) are not implemented for command WRITE_DAQ.

5.2.4.4.41. lim.Xcp.EB_INTREQ_Xcp_0041

Description	Although specified by ASAM, error codes ERR_DAQ_ACTIVE(0x11), ERR_RES_-TEMP_NOT_A(0x33) are not implemented for command SET_DAQ_LIST_MODE.

5.2.4.4.42. lim.Xcp.EB_INTREQ_Xcp_0042

Description	Although specified by ASAM, error codes ERR_MODE_NOT_VALID(0x27), ERR_-RES_TEMP_NOT_A(0x33) are not implemented for command START_STOP_DAQ_-LIST.

5.2.4.4.43. lim.Xcp.EB_INTREQ_Xcp_0043

Description	Although specified by ASAM, error codes ERR_MODE_NOT_VALID(0x27), ERR_-DAQ_CONFIG(0x2A), ERR_RES_TEMP_NOT_A(0x33) are not implemented for command START_STOP_SYNCH.

5.2.4.4.44. lim.Xcp.EB_INTREQ_Xcp_0044

Description	Although specified by ASAM, error codes ERR_MODE_NOT_VALID(0x27), ERR_-DAQ_CONFIG(0x2A), ERR_RES_TEMP_NOT_A(0x33) are not implemented for command CLEAR_DAQ_LIST.

5.2.4.4.45. lim.Xcp.EB_INTREQ_Xcp_0045

Description	Although specified by ASAM, error codes ERR_CMD_UNKNOWN(0x27), ERR_-RES_TEMP_NOT_A(0x33) are not implemented for command GET_DAQ_LIST_IN-FO.

5.2.4.4.46. lim.Xcp.EB_INTREQ_Xcp_0046

Description	Although specified by ASAM, error code ERR_RES_TEMP_NOT_A(0x33) is not im-plemented for command GET_DAQ_CLOCK.

5.2.4.4.47. lim.Xcp.EB_INTREQ_Xcp_0047

Description	Although specified by ASAM, error codes ERR_CMD_UNKNOWN(0x20), ERR_-RES_TEMP_NOT_A(0x33) are not implemented for command GET_DAQ_PRO-CESSOR_INFO.

5.2.4.4.48. lim.Xcp.EB_INTREQ_Xcp_0048

Description	Although specified by ASAM, error codes ERR_CMD_UNKNOWN(0x20), ERR_-RES_TEMP_NOT_A(0x33) are not implemented for command GET_DAQ_RESO-LUTION_INFO.

5.2.4.4.49. lim.Xcp.EB_INTREQ_Xcp_0049

Description	Although specified by ASAM, error code ERR_RES_TEMP_NOT_A(0x33) is not implemented for command GET_DAQ_LIST_MODE.

5.2.4.4.50. lim.Xcp.EB_INTREQ_Xcp_0050

Description	Although specified by ASAM, error codes ERR_CMD_UNKNOWN(0x20), ERR_-RES_TEMP_NOT_A(0x33) are not implemented for command GET_DAQ_EVENT_-INFO.

5.2.4.4.51. lim.Xcp.EB_INTREQ_Xcp_0051

Description	Although specified by ASAM, error codes ERR_CMD_UNKNOWN(0x20), ERR_-RES_TEMP_NOT_A(0x33) are not implemented for command FREE_DAQ.

5.2.4.4.52. lim.Xcp.EB_INTREQ_Xcp_0052

Description	Although specified by ASAM, error codes ERR_CMD_UNKNOWN(0x20), ERR_-RES_TEMP_NOT_A(0x33) are not implemented for command ALLOC_DAQ.

5.2.4.4.53. lim.Xcp.EB_INTREQ_Xcp_0053

Description	Although specified by ASAM, error code ERR_CMD_UNKNOWN(0x20) is not implemented for command ALLOC_DAQ.

5.2.4.4.54. lim.Xcp.EB_INTREQ_Xcp_0054

Description	Although specified by ASAM, error code ERR_CMD_UNKNOWN(0x20) is not implemented for command ALLOC_ODT_ENTRY.

5.2.4.4.55. lim.Xcp.EB_INTREQ_Xcp_0055

Description	Although specified by ASAM, error code ERR_RES_TEMP_NOT_A(0x33) is not implemented for command PROGRAM_START.

5.2.4.4.56. lim.Xcp.EB_INTREQ_Xcp_0056

Description	Although specified by ASAM, error codes ERR_MEMORY_OVERFLOW(0x30), ERR_RES_TEMP_NOT_A(0x33) are not implemented for command PROGRAM.

5.2.4.4.57. lim.Xcp.EB_INTREQ_Xcp_0057

Description	Although specified by ASAM, error code ERR_RES_TEMP_NOT_A(0x33) is not implemented for command PROGRAM.

5.2.4.4.58. lim.Xcp.EB_INTREQ_Xcp_0058

Description	Although specified by ASAM, error code ERR_RES_TEMP_NOT_A(0x33) is not implemented for command GET_PGM_PROCESSOR_INFO.

5.2.4.4.59. lim.Xcp.EB_INTREQ_Xcp_0059

Description	Although specified by ASAM, error codes ERR_CMD_UNKNOWN(0x20), ERR_OUT_OF_RANGE(0x22), ERR_ACCESS_DENIED(0x24), ERR_ACCESS_LOCKED(0x25), ERR_MEMORY_OVERFLOW(0x30), ERR_RES_TEMP_NOT_A(0x33) are not implemented for command PROGRAM_NEXT.

5.2.4.4.60. lim.Xcp.EB_INTREQ_Xcp_0060

Description	Only one buffer with CMD packet type and only one buffer with RES_ERR packet type is allowed per connection when Xcp FlexRay buffers are configured.
Rationale	This is the common use case which allows a more efficient implementation.

5.2.4.4.61. lim.Xcp.EB_INTREQ_Xcp_0061

Description	An Xcp FlexRay buffer is mapped to an L-PDU which, in turn, is mapped to a FlexRay frame. That FlexRay frame can contain only one PDU.
Rationale	Each Xcp FlexRay buffer is also mapped to a connection information to share the Packet Type information. That connection information is mapped to a single PDU, therefore we must keep the 1 to 1 connection also on FlexRay interface side.

5.2.4.4.62. lim.Xcp.EB_INTREQ_Xcp_0062

Description	The Xcp_MainFunction() periodicity must be the same as the FlexRay synchronous task periodicity.
Rationale	FlexRay interface module does not implicitly initializes the FlexRay buffers which are marked as reconfigurable (FrIfReconfigurable is true). This means that Xcp must initialize explicitly those buffers and, for that, a FlexRay SYNC state monitor mechanism is implemented. In order for that mechanims to be consistent, the Xcp_MainFunction() periodicity must be the same as the FlexRay synchronous

5.2.4.4.63. lim.Xcp.EB_INTREQ_Xcp_0063

Description	64 bit platforms are not supported.
Rationale	The Xcp protocol uses 4 byte memory addresses.

5.2.4.4.64. lim.Xcp.EB_INTREQ_Bsw_Distribution_1

Description	The integrator must ensure that no parallel communication is done with the Xcp slave. This includes any attempt to establish a connection while the Xcp slave is already processing a CONNECT command or TP commands.
Rationale	This allows for an optimal implementation with respect to cross core data access and protection.

5.2.4.4.65. lim.Xcp.EB_INTREQ_Bsw_Distribution_2

Description	Resume mode shall not be used if XcpBswDistribution is enabled.

Rationale	There is no bsw distribution support for NvM at the moment.

5.2.4.4.66. lim.Xcp.EB_INTREQ_Bsw_Distribution_StoreCall

Description	Xcp_ApplSetReqStoreCalReq shall be implemented in such a way that it can access NvM APIs.
Rationale	There is no bsw distribution support for NvM at the moment.

5.2.4.4.67. lim.Xcp.EB_INTREQ_Bsw_Distribution_Xcp_Init

Description	If XcpBswDistribution is enabled Xcp_Init shall be called on the Master Partition.
Rationale	The Xcp_Init for the other partition will be sent by Master Partition.

5.2.4.4.68. lim.Xcp.EB_INTREQ_Bsw_Distribution_NvM_Integration

Description	If XcpBswDistribution is enabled, NvM module shall be integrated on the same partition as the Xcp Master Partition.
Rationale	All Nv requests are forwarded to the Master Partition, which call NvM APIs.

5.3. XcpR

5.3.1. Configuration parameters

Containers included		
Container name	**Multiplicity**	**Description**
CommonPublishedInformation	1..1	**Label:** Common Published Information

Containers included		
		Common container, aggregated by all modules. It contains published information about vendor and versions.
PublishedInformation	1..1	**Label:** EB Published Information Additional published parameters not covered by Common-PublishedInformation container.
XcpRDefensiveProgramming	1..1	**Label:** Defensive Programming Options Parameters for defensive programming
XcpRGeneral	1..1	

Parameters included	
Parameter name	**Multiplicity**
IMPLEMENTATION_CONFIG_VARIANT	1..1

Parameter Name	IMPLEMENTATION_CONFIG_VARIANT	
Label	Config Variant	
Multiplicity	1..1	
Type	ENUMERATION	
Default value	VariantPreCompile	
Range	VariantPreCompile	
Configuration class	**VariantPreCompile:**	VariantPreCompile

5.3.1.1. CommonPublishedInformation

Parameters included	
Parameter name	**Multiplicity**
ArMajorVersion	1..1
ArMinorVersion	1..1
ArPatchVersion	1..1
SwMajorVersion	1..1
SwMinorVersion	1..1
SwPatchVersion	1..1
ModuleId	1..1

Parameters included	
VendorId	1..1
Release	1..1

Parameter Name	ArMajorVersion
Label	AUTOSAR Major Version
Description	Major version number of AUTOSAR specification on which the appropriate implementation is based on.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	0
Configuration class	PublishedInformation:
Origin	Elektrobit Automotive GmbH

Parameter Name	ArMinorVersion
Label	AUTOSAR Minor Version
Description	Minor version number of AUTOSAR specification on which the appropriate implementation is based on.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	0
Configuration class	PublishedInformation:
Origin	Elektrobit Automotive GmbH

Parameter Name	ArPatchVersion
Label	AUTOSAR Patch Version
Description	Patch level version number of AUTOSAR specification on which the appropriate implementation is based on.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	0
Configuration class	PublishedInformation:
Origin	Elektrobit Automotive GmbH

Parameter Name	SwMajorVersion

Label	Software Major Version
Description	Major version number of the vendor specific implementation of the module.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	1
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**SwMinorVersion**
Label	Software Minor Version
Description	Minor version number of the vendor specific implementation of the module. The numbering is vendor specific.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	1
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**SwPatchVersion**
Label	Software Patch Version
Description	Patch level version number of the vendor specific implementation of the module. The numbering is vendor specific.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	1
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**ModuleId**
Label	Numeric Module ID
Description	Module ID of this module from Module List
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	255

Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**VendorId**
Label	Vendor ID
Description	Vendor ID of the dedicated implementation of this module according to the AUTOSAR vendor list
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	1
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**Release**
Label	Release Information
Multiplicity	1..1
Type	STRING_LABEL
Default value	
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

5.3.1.2. PublishedInformation

Parameters included	
Parameter name	**Multiplicity**
PbcfgMSupport	1..1

Parameter Name	**PbcfgMSupport**
Label	PbcfgM support
Description	Specifies whether or not the XcpR can use the PbcfgM module for post-build support.
Multiplicity	1..1
Type	BOOLEAN
Default value	false

Configuration class	**PublishedInformation:**	
Origin	Elektrobit Automotive GmbH	

5.3.1.3. XcpRDefensiveProgramming

Parameters included	
Parameter name	**Multiplicity**
XcpRDefProgEnabled	1..1
XcpRPrecondAssertEnabled	1..1
XcpRPostcondAssertEnabled	1..1
XcpRStaticAssertEnabled	1..1
XcpRUnreachAssertEnabled	1..1
XcpRInvariantAssertEnabled	1..1

Parameter Name	**XcpRDefProgEnabled**	
Label	Enable Defensive Programming	
Description	Enables or disables the defensive programming feature for the module XcpR. Note: This feature is dependent on the use of the development error detection module. To use the defensive programming feature, proceed as follows: 1. Enable development error detection 2. Enable defensive programming 3. Enable assertions as required	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**XcpRPrecondAssertEnabled**
Label	Enable Precondition Assertions
Description	Enables handling of precondition assertion checks reported from the module XcpR. Dependency on parameter(s):

	▶ Enable Development Error Detection (`XcpRDevErrorDetect`): must be enabled ▶ Enable Defensive Programming (`XcpRDefProgEnabled`): must be enabled
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpRPostcondAssertEnabled**
Label	Enable Postcondition Assertions
Description	Enables handling of postcondition assertion checks reported from the module XcpR. Dependency on parameter(s): ▶ Enable Development Error Detection (`XcpRDevErrorDetect`): must be enabled ▶ Enable Defensive Programming (`XcpRDefProgEnabled`): must be enabled
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpRStaticAssertEnabled**
Label	Enable Static Assertions
Description	Enables handling of static assertion checks reported from the module XcpR. Dependency on parameter(s): ▶ Enable Development Error Detection (`XcpRDevErrorDetect`): must be enabled ▶ Enable Defensive Programming (`XcpRDefProgEnabled`): must be enabled
Multiplicity	1..1

Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**XcpRUnreachAssertEnabled**	
Label	Enable Unreachable Code Assertions	
Description	Enables handling of unreachable code assertion checks reported from the module XcpR. Dependency on parameter(s): ► Enable Development Error Detection (`XcpRDevErrorDetect`): must be enabled ► Enable Defensive Programming (`XcpRDefProgEnabled`): must be enabled	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**XcpRInvariantAssertEnabled**	
Label	Enable Invariant Assertions	
Description	Enables handling of invariant assertion checks reported from functions of the module XcpR. Dependency on parameter(s): ► Enable Development Error Detection (`XcpRDevErrorDetect`): must be enabled ► Enable Defensive Programming (`XcpRDefProgEnabled`): must be enabled	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPreCompile:**	VariantPreCompile

Origin	Elektrobit Automotive GmbH

5.3.1.4. XcpRGeneral

Containers included		
Container name	**Multiplicity**	**Description**
XcpRSourcePDUConfiguration	1..n	Groups PDU information related to a connection via a specified bus(Can/CanFD/FlexRay/Ethernet/CDD) Multiple connection via the same bus type can be configured Ethernet source connections are not supported in the current implementation of the XcpR
XcpRDestinationPDUConfiguration	1..n	Groups PDU information related to a destination connection via a specified bus(Can/CanFD/FlexRay/Ethernet) Multiple connection via the same bus type can be configured Ethernet destination connections are not supported in the current implementation of the XcpR
XcpRRoutingPaths	1..n	**Label:** XcpR PDU Mapping Defines PDU information. A PDU may be either a transmission (Tx) PDU or a reception (Rx) PDU.
XcpRRoutingToConnectionMapping	0..n	**Label:** XcpR Connection Mapping List of Connection Groups, which logically groups configured Routing Paths. Each connection group shall be composed of at least: ► A routing path for requests from a master Xcp to a slave Xcp. ► A routing path for the responses from the slave Xcp to the master Xcp. Additional routing paths can be mapped to the group for DAQ/STIM. If the list is empty, the "Connection Group" functionality is disabled.

Parameters included	
Parameter name	**Multiplicity**
XcpRMultipleSlavesEnabled	1..1
XcpRDevErrorDetect	1..1
XcpRDetRuntimeChecks	1..1
XcpRVersionInfoApi	1..1
XcpRMainFunctionPeriod	1..1
XcpRQueueSize	1..1
XcpRTxRetryCount	1..1
XcpRTxBusRetry	1..1
XcpRTxBusTimeout	1..1
XcpREventPacketEnabled	1..1

Parameter Name	**XcpRMultipleSlavesEnabled**	
Label	XcpR support for multiple Xcp slaves	
Description	If enabled, the CONNECT command can be used either in MODE=0 to connect to the XCP instance on the local ECU (Modulator) or in MODE=1 to connect to the Xcp Slave on the remote ECU. If enabled, two XcpR destinations can be configured for an XcpR source (which is enabled as the connection to the master Xcp). If disabled, only one XcpR destination for an XcpR source (which is enabled as the connection to the master Xcp, can be configured.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**XcpRDevErrorDetect**
Label	Development Error Detection
Description	Enables the detection of development errors. **Optimization Effect:** ► **ROM reduction (code):** Disabling this parameter reduces the ROM consumption of the module code.

	► **Execution time reduction(code):** Disabling this parameter reduces the execution time of the module code.
Multiplicity	1..1
Type	BOOLEAN
Default value	true

Parameter Name	**XcpRDetRuntimeChecks**	
Label	DET runtime checks	
Description	Enables or disables the reporting of Default Error Trace runtime checks for the module XcpR via Det_ReportRuntimeError() and Det_ReportTransientFault(). Note: This parameter enables/disables the reporting. The runtime check itself won't be disabled. ► True: Reporting of runtime checks enabled ► False: Reporting of runtime checks disabled	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	true	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**XcpRVersionInfoApi**
Label	Version Info API
Description	Enables the version information API (`XcpR_GetVersionInfo()`) **Optimization Effect:** ► **ROM reduction (code):** Disabling this parameter reduces the ROM consumption of the module code.
Multiplicity	1..1
Type	BOOLEAN
Default value	false

Parameter Name	**XcpRMainFunctionPeriod**
Label	Main Function Period [s]
Description	Defines the time interval in which the XcpR Main function will be invoked by BSW Scheduler.

	The XcpR_MainFunction() will process receptions and transmissions of XcpR PDUs in case reception/transmission from RxIndication/TxConfirmation is not enabled. If the "multiple packages in one XCPR frame" option is configured, those packages will be processed during execution of XcpR_MainFunction(). Range: ► 0.001 .. 255
Multiplicity	1..1
Type	FLOAT
Default value	0.005

Parameter Name	**XcpRQueueSize**	
Label	Queue size for storing received XcpR messages	
Description	Defines the queue size for the XcpR internal ring-buffer. Messages received by XcpR are copied in an internal buffer after they are stripped of the header or other bus-specific information, in order to be adapted afterwards to the configured destination bus type *Please note that this parameter might have a major impact on the resource consumption of your project*. Take into consideration that for each stored message, there are additional internal admin data also stored. This internal data is composed of: 1 PduLengthType, 1 PduIdType and 1 additional byte. For example, if XcpRQueueSize = 260 and PduLengthType, PduIdType are uint16, the XcpR internal queue can store a maximum of 20 messages of 8 bytes each. ((8 bytes for a message + 5 bytes of internal data) * 20) = 260 bytes.	
Multiplicity	1..1	
Type	INTEGER	
Default value	128	
Range	<=65535 >=0	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**XcpRTxRetryCount**

Label	Tx Retry Count
Description	Defines the number of times the data will be retried for transmission. Range: ► 0 .. 255 **Optimization Effect:** ► **ROM reduction (code):** Setting the value ZERO for this parameter reduces the ROM consumption of the module code. ► **RAM reduction (code):** Setting the value ZERO for this parameter reduces the RAM consumption of the module code.
Multiplicity	1..1
Type	INTEGER
Default value	0
Range	<=255 >=0
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpRTxBusRetry**
Label	Tx Bus Retry
Description	Defines the number of retries the XcpR performs in order to transmit a Pdu. Whenever a PDU is accepted to be transmitted by the lower layer, XcpR waits for `XcpRTxBusTimeout` time to receive the confirmation of successful transmission. If the timeout occurs, XcpR retries the whole lost PDU transmission for the configured number of times. When the number of retries expires (if it is not configured with the reserved value of 255 - infinite retries), XcpR autonomously disconnects from the connection. Range: ► 0 .. 254 ► 255: reserved value for infinite number of retries Note: Not to be confused with `XcpRTxRetryCount` which retries only the transmit request rejected by the interface layer (i.e. calls to `xxxIf_Transmit()`). **Optimization Effect:**

	► **ROM reduction (code):** Disabling this parameter reduces the ROM consumption of the module code. ► **RAM reduction (code):** Disabling this parameter reduces the RAM consumption of the module code.
Multiplicity	1..1
Type	INTEGER
Default value	0
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpRTxBusTimeout**
Label	Tx Bus Timeout[s]
Description	Defines the maximum allowed time frame for a message to be transmitted from the XcpR to the remote Xcp Slave or Xcp Master. If no retries (`XcpRTxBusRetry`) are configured, when the timeout occurs, for the connection to the Xcp Master, the XcpR will autonomously disconnect. Otherwise, the message will be re-transmitted for the configured ammounts of retries, each re-transmission attempt will reset the value of the internal counter to this configured timeout value. Range: ► 0.001 .. 255
Multiplicity	1..1
Type	FLOAT
Default value	2.0
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpREventPacketEnabled**
Label	Event (EV) Packet Support
Description	Enables the transmission of EV_SESSION_TERMINATED from the XcpR to the master device.
Multiplicity	1..1
Type	BOOLEAN
Default value	true

Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

5.3.1.5. XcpRSourcePDUConfiguration

Containers included		
Container name	**Multiplicity**	**Description**
XcpRSourceInterfaceType	1..1	**Label:** XcpR Source Interface Type Choice container used to select the source connection type: ► `XcpRSourceConnectionOverCAN`: only PDUs mapped to the referenced CanIf/Configuration container the selected connection configuration ► `XcpRSourceConnectionOverCANFD`: only PDUs mapped to the referenced CanIf/Configuration container the selected connection configuration ► `XcpRSourceConnectionOverFlexRay`: only PDUs mapped to the same FlexRay controller are accepted to the selected connection configuration. ► `XcpRSourceConnectionOverEthernet`: only PDUs mapped to the same Socket connection group are accepted to the selected connection configuration. ► `XcpRSourceConnectionOverCdd`: only PDUs referenced from a CDD configuration are accepted to the selected connection configuration.
XcpRPdu	1..1	**Label:** XcpR PDU Type Defines PDU information. A PDU may be either a transmission (Tx) PDU or a reception (Rx) PDU.
XcpRSourcePDUAttributes	1..1	PDU attributes for each configured XcpR Source.

Parameters included	
Parameter name	**Multiplicity**
XcpRIsSrcMasterConnection	1..1

Parameter Name	**XcpRIsSrcMasterConnection**

Label	Enable this container as a source from the master Xcp	
Description	Defines this source as the connection to the master Xcp. Only a Rx source can be defined as master.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**PreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

5.3.1.6. XcpRSourceInterfaceType

Containers included		
Container name	**Multiplicity**	**Description**
XcpRSourceConnectionOver-CAN	1..1	
XcpRSourceConnectionOver-CANFD	1..1	
XcpRSourceConnectionOver-FlexRay	1..1	
XcpRSourceConnec-tionOverEthernet	1..1	
XcpRSourceConnectionOver-CDD	1..1	

5.3.1.7. XcpRSourceConnectionOverCAN

Parameters included	
Parameter name	**Multiplicity**
XcpRSourceConnectionCanIfCfgRef	1..1

Parameter Name	**XcpRSourceConnectionCanIfCfgRef**
Label	Reference to CanIf configuration
Description	Reference to the CanIf interface where the Source/Destination PDUs, configured for the selected connection, can be found

	This reference is used to validate/generate PDUs information for the assigned PDUs for the selected connection over CAN. Dependency on parameter(s): ► XcpR source interface type (`XcpRSourceInterfaceType`): The selected value for this parameter has to be `XcpRSourceConnectionOverCAN`. Otherwise this parameter is not used. ► A valid CanIf configuration has to be avaible for selection.
Multiplicity	1..1
Type	REFERENCE
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

5.3.1.8. XcpRSourceConnectionOverCANFD

Parameters included	
Parameter name	**Multiplicity**
XcpRSourceCanFdMaxDlcRequired	1..1
XcpRSourceCanFdMaxDlc	1..1
XcpRSourceConnectionCanFdCanIfCfgRef	1..1

Parameter Name	**XcpRSourceCanFdMaxDlcRequired**
Label	CANFD Max DLC required
Description	Defines if the length of the XcpR packages is given by `XcpRSourceCanFdMaxDlc` or not. For CAN-FD, the next larger defined DLC is considered for the calculation if `XcpRSourceCanFdMaxDlcRequired` is disabled and DLC of a frame is not matching one of the defined DLC values. If `XcpRSourceCanFdMaxDlcRequired` is enabled, always the corresponding DLC = `XcpRSourceCanFdMaxDlc` parameter is used for the calculation.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpRSourceCanFdMaxDlc**

Label	CAN-FD Max DLC
Description	Defines the value of maximum data length of a CAN-FD frame. The maximum data length of a CAN-FD message shall either one of the following values 8,12,16,20,24,32,48,64. Range: ► DLC values: 8,12,16,20,24,32,48,64
Multiplicity	1..1
Type	INTEGER
Default value	64
Range	8 12 16 20 24 32 48 64
Configuration class	**PreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpRSourceConnectionCanFdCanIfCfgRef**
Label	Reference CAN-FD to CanIf configuration
Description	Reference to the CanIf interface where the Source/Destination PDUs, configured for the selected connection, can be found This reference is used to validate/generate PDUs information for the assigned PDUs for the selected connection over CAN-FD. Dependency on parameter(s): ► XcpR connection type (`XcpRSourceInterfaceType`): The selected value for this parameter has to be `XcpRSourceConnectionOverCANFD`. Otherwise this parameter is not used. ► A valid CanIf configuration has to be avaible for selection.
Multiplicity	1..1

Type	REFERENCE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

5.3.1.9. XcpRSourceConnectionOverFlexRay

Parameters included	
Parameter name	**Multiplicity**
XcpRSourceFlxNodeAddress	1..1
XcpRSourceFlxHeaderAlignment	1..1
XcpRSourcePackMultiMsgInOneFlexRayFrame	1..1
XcpRSourceSequenceCorrectionEnabled	1..1
XcpRSourceMaxFlexMsgLength	1..1
XcpRSourceConnectionFrIfCfgRef	1..1

Parameter Name	**XcpRSourceFlxNodeAddress**	
Label	Node Address for XcpR (NAX)	
Description	Defines the node address for XcpR (NAX) that is used for this node. This value will be written to the NAX field of the FlexRay XcpR header when a message is expected in messages received from the master/slave in the NAX field. When multiple FlexRay are configured, it is allowed to share the same NAX address between connections, or to have different NAX address. Range: ► 0 .. 255	
Multiplicity	1..1	
Type	INTEGER	
Default value	0	
Range	<=255 >=0	
Configuration class	**VariantPreCompile:**	VariantPreCompile

Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpRSourceFlxHeaderAlignment**
Label	Alignment of the FlexRay XcpR Header
Description	Defines the alignment of the FlexRay XcpR header. Range: ► PACKET_ALIGNMENT_8: 8 bit alignment is used. ► PACKET_ALIGNMENT_16: 16 bit alignment is used. ► PACKET_ALIGNMENT_32: 32 bit alignment is used. **Optimization Effect:** ► **RAM reduction (code):** Selecting a smaller value for this parameter reduces the RAM consumption of the module code.
Multiplicity	1..1
Type	ENUMERATION
Default value	PACKET_ALIGNMENT_8
Range	PACKET_ALIGNMENT_8 PACKET_ALIGNMENT_16 PACKET_ALIGNMENT_32
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpRSourcePackMultiMsgInOneFlexRayFrame**
Label	Enable Multiple XcpR messages In One FlexRay Frame
Description	Enables the concatenation of multiple XcpR messages in one frame, for FlexRay communication
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpRSourceSequenceCorrectionEnabled**
Label	Enable Sequence Correctness

Description	Enables the sequence correction. Each sent frame will contain a Counter that is incremented for each XcpR packet.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**XcpRSourceMaxFlexMsgLength**
Label	Max Flexray Msg Length
Description	Defines the maximum data length of a FlexRay frame (*MAX_FLX_LEN_BUF_x_init*) that XcpR is able to receive in a specific XcpR-dedicated buffer. The length of the message can be variable. However the maximum length of an XcpR on FlexRay message is *MAX_FLX_LEN_ABS* = 254 bytes. *MAX_FLX_LEN_BUF_x* is the actual maximum data length of a FlexRay frame in a specific buffer and which can be non-configurable.Thus making *MAX_FLX_LEN_BUF_x* always equal to *MAX_FLX_LEN_BUF_x_init*. Present implementation uses the non-configurable *MAX_FLX_LEN_BUF_x*. Thus for present implementation *MAX_FLX_LEN_BUF_x* equals *MAX_FLX_LEN_BUF_x_init* for all buffers. Range: ► 8 .. 254 Dependency on other parameter(s): This parameter is available if all of the following conditions are fulfilled: ► Multiple XcpR Msgs In One XcpR FlexRay Frame (`XcpRSourcePackMultiMsgInOneFlexRayFrame`) is enabled. Hint: this parameter should be high enough to be able to accomodate at least 2 maximum size DTOs
Multiplicity	1..1
Type	INTEGER
Default value	254
Configuration class	**PreCompile:** VariantPreCompile

Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpRSourceConnectionFrIfCfgRef**	
Label	Reference to FrIf configuration	
Description	Reference to the FrIf interface where the Source/Destination PDUs, configured for the selected connection, can be found This reference is used to validate/generate PDUs information for the assigned PDUs for the selected source connection over FlexRay. Dependency on parameter(s): ► XcpR connection type (`XcpRSourceInterfaceType`): The selected value for this parameter has to be `XcpRSourceConnectionOverFlexRay`. Otherwise this parameter is not used. ► A valid FrIf configuration has to be avaible for selection.	
Multiplicity	1..1	
Type	REFERENCE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

5.3.1.10. XcpRSourceConnectionOverEthernet

Parameters included	
Parameter name	**Multiplicity**
XcpRSourceOpenSoCon	1..1
XcpRSourcePackMultiMsgInOneEthernetFrame	1..1
XcpRSourceMaxEthernetMsgLength	1..1
XcpRSourceConnectionSoAdConfigRef	1..1

Parameter Name	**XcpRSourceOpenSoCon**
Label	Open the Socket Connection
Description	Enables XcpR to handle opening of SoAd sockets(s) during startup. If enabled `SoAd_OpenSoCon()` will be called during `XcpR_Init()` for each configured socket. If UDP protocol is used, this parameter must be enabled.

	Note: Please ensure that `SoAd_Init()` will be called before `XcpR_Init()`. Dependency on parameter(s): ► XcpR connection type (`XcpRSourceInterfaceType`): The selected value for this parameter has to be `XcpRSourceConnectionOverEthernet`. Otherwise this parameter is not used.
Multiplicity	1..1
Type	BOOLEAN
Default value	true
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpRSourcePackMultiMsgInOneEthernetFrame**
Label	Enable Multiple XcpR messages In One Ethernet Frame
Description	Enables the concatenation of multiple XcpR messages in one frame, for Ethernet communication. Note: Caution must be taken when enabling this feature, in case the selected communication type is TCP/IP, because the Master should be able to support the packing of multiple pdus in one frame, for TCP/IP, too. Rationale: In ASAM specifications only for UDP/IP is specified that the "Suport for multiple PDUs in one frame may be implemented"
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpRSourceMaxEthernetMsgLength**
Label	Max Eternet Msg Length
Description	Defines the maximum data length of a Ethernet frame that a specific slave is able to receive or transmit in a frame. It is measured in bytes The length of the message can be variable. However the maximum length of an XcpR on Eternet message is limited to the Ethernet payload length limits in bytes. Range:

	► 8 .. 1454 Dependency on other parameter(s): This parameter is available only if both of the following parameters are enabled: ► XcpR on Ethernet (`XcpROnEthernetEnabled`). ► Multiple XcpR Msgs In One XcpR Frame (`XcpRPackMultiMsgInOneEthernetFrame`). Default value for this parameter is calculated to always be able to accomodate 1 maximum size CTO plus 5 maximum size DTOs Hint this parameter should be high enough to be able to accomodate at least 2 maximum size DTOs
Multiplicity	1..1
Type	INTEGER
Default value	1454
Configuration class	**PreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpRSourceConnectionSoAdConfigRef**
Label	Reference to SoAd configuration
Description	Reference to the SoAd interface where the Source/Destination PDUs, configured for the selected connection, can be found This reference is used to validate/generate PDUs information for the assigned PDUs for the selected connection over Ethernet. Dependency on parameter(s): ► XcpR connection type (`XcpRSourceConnectionInterfaceType`): The selected value for this parameter has to be `XcpRSourceConnectionOverEthernet`. Otherwise this parameter is not used. ► A valid SoAd configuration has to be avaible for selection. ► One entry with the name 'XcpR', containing the UL configuration for XcpR module in SoAd, has to be added into the `SoAdBswModules` container. ► `SoAdIf` has to be enabled, for the dedicated XcpR configuration entry into `SoAdBswModules`. ► `SoAdTp` has to be disabled, for the dedicated XcpR configuration entry into `SoAdBswModules`

	▶ SoAdIfTxConfirmation has to be enabled, for the dedicated XcpR configuration entry into SoAdBswModules entry. ▶ SoAdLocalIpAddrAssigmentChg has to be disabled, for the dedicated XcpR configuration entry into SoAdBswModules(refer to ASAM specifications). ▶ SoAdIfTriggerTransmit has to be disabled , for the dedicated XcpR configuration entry into SoAdBswModules(the trigger transmit api is not supported for Ethernet). ▶ SoAdSoConModeChg parameter, configured for the dedicated XcpR configuration entry from SoAdBswModules container, has to be:. ▶ Enabled, if there is configured at least one XcpR connection over TCP/IP. ▶ Disabled, if there is no XcpR connection over TCP/IP, configured.
Multiplicity	1..1
Type	REFERENCE
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

5.3.1.11. XcpRSourceConnectionOverCDD

5.3.1.12. XcpRPdu

Containers included		
Container name	**Multiplicity**	**Description**
XcpRRxPdu	1..1	Defines the Rx PDUs.
XcpRTxPdu	1..1	Defines the Tx PDU.

5.3.1.13. XcpRRxPdu

Parameters included	
Parameter name	**Multiplicity**

Parameters included	
XcpRRxSourcePduReference	1..1
XcpRRxSourcePduId	1..1
XcpRSrcPduSupportRxFromRxIndication	1..1
XcpRRxSrcPduSupportTxFromRxIndication	1..1

Parameter Name	XcpRRxSourcePduReference	
Label	Reference to Rx source PDU	
Description	Reference to the external Rx PDU definition in the EcuC module. Dependency on parameter(s): ► A valid reference to EcuC's `Pdu` parameters has to be provided. ► A unique EcuC PDU has to be provided as reference for each XcpR Rx Pdu. ► It is not allowed to configure bidirectional XcpR PDUs(the same PDU cannot be given as reference for both `XcpRRxPdu` and `XcpRTxPdu`).	
Multiplicity	1..1	
Type	REFERENCE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	XcpRRxSourcePduId	
Label	XcpRRxSourcePduId	
Description	Defines the ID of the PDU that will be received via a XcpR_RxIndication(). The XcpR's Rx Pdu ids has to be zero based and consecutive. Range: 0 .. 65535	
Multiplicity	1..1	
Type	INTEGER	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	XcpRSrcPduSupportRxFromRxIndication
Label	Support for reception from RxIndication

Description	Support for reception from Rx indication. If enabled a reception for this PDU will be triggered when XcpR_RxIndication() is executed. **Enabling this feature for several PDUs might have impact on the run-time by creating a big overhead in the system.** Dependency on parameter(s): none.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**XcpRRxSrcPduSupportTxFromRxIndication**	
Label	Support for transmission from RxIndication	
Description	Support for transmission from Rx indication. If reception from RxIndication is enabled, this parameter can also be enabled to trigger a transmission of the received message in the context of XcpR_RxIndication(). If this parameter is disabled, the transmission of the message received in the context of RxIndication will be done in the context of XcpR_MainFunction(). **Enabling this feature for several PDUs might have impact on the run-time by creating a big overhead in the system.** Dependency on parameter(s): XcpRSrcPduSupportRxFromRxIndication (Support for reception from RxIndication).	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**PreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

5.3.1.14. XcpRTxPdu

Parameters included	
Parameter name	**Multiplicity**
XcpRTxSourcePduReference	1..1

Parameters included	
XcpRTxSourcePduId	1..1
XcpRSrcPduSupportRxFromXcpRTransmit	1..1
XcpRTxSrcPduSupportTxFromXcpRTransmit	1..1

Parameter Name	XcpRTxSourcePduReference	
Label	Reference to Tx source PDU	
Description	Reference to the external Tx PDU definition in the EcuC module. Dependency on parameter(s): ► A valid reference to EcuC's `Pdu` parameters has to be provided. ► A unique EcuC PDU has to be provided as reference for each XcpR Tx Pdu. ► It is not allowed to configured bidirectional XcpR PDUs(the same PDU cannot be given as reference for both `XcpRTxPdu` and `XcpRRxPdu`).	
Multiplicity	1..1	
Type	REFERENCE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	XcpRTxSourcePduId	
Label	XcpRTxSourcePduId	
Description	Defines the PDU identifier, which has to be used by the upper layer BSW module for XcpR_Transmit(). The XcpR's Tx Pdu ids has to be zero based and consecutive. Range: 0 .. 65535	
Multiplicity	1..1	
Type	INTEGER	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	XcpRSrcPduSupportRxFromXcpRTransmit
Label	Support for reception from XcpR_Transmit()
Description	Support for reception from XcpR_Transmit(). If enabled a reception for this PDU will be triggered when XcpR_Transmit() is executed.

	Enabling this feature for several PDUs might have impact on the run-time by creating a big overhead in the system. Dependency on parameter(s): none.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpRTxSrcPduSupportTxFromXcpRTransmit**
Label	Support for transmission from XcpR_Transmit()
Description	Support for transmission from XcpR_Transmit(). If reception from XcpR_Transmit() is enabled, this parameter can also be enabled to trigger a transmission of the received message in the context of XcpR_Transmit(). If this parameter is disabled, the transmission of the message received in the context of XcpR_Transmit() will be done in the context of XcpR_MainFunction(). **Enabling this feature for several PDUs might have impact on the run-time by creating a big overhead in the system.** Dependency on parameter(s): XcpRSrcPduSupportRxFromXcpRTransmit (Support for reception from XcpR_Transmit()).
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**PreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

5.3.1.15. XcpRSourcePDUAttributes

Parameters included	
Parameter name	**Multiplicity**
XcpRSrcMaxCto	1..1
XcpRSrcMaxDto	1..1

Parameters included	
XcpRSrcMaxCtoPgm	0..1
XcpRSrcTimestampType	1..1

Parameter Name	XcpRSrcMaxCto	
Label	Maximum CTO Length	
Description	The maximum length of a CTO packet in bytes, specific to each individual source connection. A CTO packet consists of: ► An Identification Field (the PID containing the CTO Packet code) ► And the Data Field which contains the specific payload necessary for the different types of CTO packet. The length of the CTO varies according to the needs and the used XCP Transport Layer. Range: ► CAN : 8 ► CAN-FD : 8..(maximum data length of a CAN-FD frame ► FlexRay : 8..(maximum data length of a FlexRay frame (`XcpRSourceMaxFlexMsgLength`) - FlexRay header length) ► Ethernet : 8..(maximum data length of a Ethernet frame (`XcpRSourceMaxEthernetMsgLength`) - Ethernet header length) **Optimization Effect:** ► **RAM reduction (code):** Selecting a smaller value for this parameter reduces the RAM consumption of the module code.	
Multiplicity	1..1	
Type	INTEGER	
Default value	8	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	XcpRSrcMaxDto
Label	Maximum DTO Length

Description	The maximum length of a DTO packet in bytes, specific to each individual source connection. A DTO packet consists of: ► An Identification Field which varies depending upon the Identification Field Type ► An optional Timestamp Field which varies depending upon the Timestamp Field Type(`XcpRSrcTimestampType`) ► And the Data Field which contains the data for synchronous acquisition and stimulation. The length of the CTO varies according to the needs and the used XCP Transport Layer. Range: ► CAN : 8 ► CAN-FD : 8..(maximum data length of a CAN-FD frame ► FlexRay : 8..(maximum data length of a FlexRay frame (`XcpRSourceMaxFlexMsgLength`) - FlexRay header length) ► Ethernet : 8..(maximum data length of a Ethernet frame (`XcpRSourceMaxEthernetMsgLength`) - Ethernet header length) **Optimization Effect:** ► **RAM reduction (code):** Selecting a smaller value for this parameter reduces the RAM consumption of the module code. ► **RAM reduction (config):** Selecting a smaller value for this parameter reduces the RAM consumption of the module config. ► **ROM reduction (config):** Selecting a smaller value for this parameter reduces the ROM consumption of the module config.
Multiplicity	1..1
Type	INTEGER
Default value	8
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpRSrcMaxCtoPgm**
Label	XcpRMaxCtoPgm

Description	Defines the maximum length of command transfer objects (CTO) in bytes for flash programming.(ASAM parameter: `MAX_CTO_PGM`) A CTO packet consists of: ► An Identification Field (the PID containing the CTO Packet code) ► And the Data Field which contains the specific payload necessary for the different types of CTO packet. The length of the CTO varies according to the needs and the used XCP Transport Layer. Range: ► CAN : 8 ► CAN-FD : 8..(maximum data length of a CAN-FD frame ► FlexRay : 8..(maximum data length of a FlexRay frame `XcpRSource-MaxFlexMsgLength` - FlexRay header length) Note: Some older Xcp masters ignore this parameter. If your master is affected, it is advised to configure this parameter to the same value as XcpRSrcMaxCto. **Optimization Effect:** ► **RAM reduction (code):** Selecting a smaller value for this parameter reduces the RAM consumption of the module code.
Multiplicity	0..1
Type	INTEGER
Default value	8
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpRSrcTimestampType**
Label	Timestamp Type
Description	Defines the type of timestamp XcpR Source is using. Range: ► NO_TIME_STAMP: no timestamp is used. ► ONE_BYTE: timestamp on 1 byte is used. ► TWO_BYTE: timestamp on 2 bytes is used.

	▶ FOUR_BYTE: timestamp on 4 bytes is used. This parameter must be set with the same value selected for the XcpTimestamp-Type of the Xcp which is connected through this routing path. Every destination that is configured as a path for this source, must have the same value for the parameter XcpRDestTimestampType.
Multiplicity	1..1
Type	ENUMERATION
Default value	NO_TIME_STAMP
Range	NO_TIME_STAMP ONE_BYTE TWO_BYTE FOUR_BYTE
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

5.3.1.16. XcpRDestinationPDUConfiguration

Containers included		
Container name	**Multiplicity**	**Description**
XcpRDestinationInterface-Type	1..1	**Label:** XcpR Destination Interface Type Choice container used to select the connection type: ▶ `XcpRDestinationConnectionOverCAN`: only PDUs mapped to the referenced CanIf/Configuration container the selected connection configuration ▶ `XcpRDestinationConnectionOverCANFD`: only PDUs mapped to the referenced CanIf/Configuration container the selected connection configuration ▶ `XcpRDestinationConnectionOverFlexRay`: only PDUs mapped to the same FlexRay controller are accepted to the selected connection configuration. ▶ `XcpRDestinationConnectionOverEthernet`: only PDUs mapped to the same Socket connection group are accepted to the selected connection configuration.

Containers included		
		▶ `XcpRDestinationConnectionOverCdd`: only PDUs referenced from a CDD configuration are accepted to the selected connection configuration.
XcpRPdu	1..1	**Label:** XcpR PDU Type Defines PDU information. A PDU may be either a transmission (Tx) PDU or a reception (Rx) PDU.
XcpRDestUpperLayerInformation	1..1	**Label:** Upper Layer configuration
XcpRDestinationPDUAttributes	1..1	PDU attributes for each configured XcpR destination.

Parameters included	
Parameter name	**Multiplicity**
XcpRIsDestMasterConnection	1..1

Parameter Name	XcpRIsDestMasterConnection	
Label	Enable this container as a destination to the master Xcp	
Description	Defines this destination as the connection to the master Xcp. Only a Tx destination can be defined as master.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**PreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

5.3.1.17. XcpRDestinationInterfaceType

Containers included		
Container name	**Multiplicity**	**Description**
XcpRDestinationConnectionOverCAN	1..1	
XcpRDestinationConnectionOverCANFD	1..1	

Containers included		
XcpRDestinationConnectionOverFlexRay	1..1	
XcpRDestinationConnectionOverEthernet	1..1	
XcpRDestinationConnectionOverCDD	1..1	

5.3.1.18. XcpRDestinationConnectionOverCAN

Parameters included	
Parameter name	**Multiplicity**
XcpRDestinationConnectionCanIfCfgRef	1..1

Parameter Name	XcpRDestinationConnectionCanIfCfgRef	
Label	Reference to CanIf configuration	
Description	Reference to the CanIf interface where the Source/Destination PDUs, configured for the selected connection, can be found This reference is used to validate/generate PDUs information for the assigned PDUs for the selected connection over CAN. Dependency on parameter(s): ► XcpR destination interface type (`XcpRDestinationInterfaceType`): The selected value for this parameter has to be `XcpRDestinationOverCAN`. Otherwise this parameter is not used. ► A valid CanIf configuration has to be avaible for selection.	
Multiplicity	1..1	
Type	REFERENCE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

5.3.1.19. XcpRDestinationConnectionOverCANFD

Parameters included	
Parameter name	**Multiplicity**

Parameters included	
XcpRDestinationCanFdMaxDlcRequired	1..1
XcpRDestinationCanFdMaxDlc	1..1
XcpRDestinationCanFdFillValue	1..1
XcpRDestinationConnectionCanFdCanIfCfgRef	1..1

Parameter Name	XcpRDestinationCanFdMaxDlcRequired	
Label	CANFD Max DLC required	
Description	Defines if the length of the XcpR packages is given by `XcpRDestinationCanFdMaxDlc` or not. For CAN-FD, the next larger defined DLC is considered for the calculation if `XcpRDestinationCanFdMaxDlcRequired` is disabled and DLC of a frame is not matching one of the defined DLC values. If `XcpRDestinationCanFdMaxDlcRequired` is enabled, always the corresponding DLC = `XcpRDestinationCanFdMaxDlc` parameter is used for the calculation.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	XcpRDestinationCanFdMaxDlc
Label	CAN-FD Max DLC
Description	Defines the value of maximum data length of a CAN-FD frame. The maximum data length of a CAN-FD message shall either one of the following values 8,12,16,20,24,32,48,64. Range: ► DLC values: 8,12,16,20,24,32,48,64
Multiplicity	1..1
Type	INTEGER
Default value	64
Range	8 12 16 20

	24	
	32	
	48	
	64	
Configuration class	**PreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**XcpRDestinationCanFdFillValue**	
Label	CAN-FD Fill Byte Value	
Description	Defines value used for the fill bytes of CANFD packages. This parameter shall be used to define the value for fill bytes. For CAN-FD fill bytes will be added to the data until the expected package length is reached. Range: ► Fill bytes Value: 0 .. 255	
Multiplicity	1..1	
Type	INTEGER	
Default value	255	
Range	<=255	
	>=0	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**XcpRDestinationConnectionCanFdCanIfCfgRef**
Label	Reference CAN-FD to CanIf configuration
Description	Reference to the CanIf interface where the Source/Destination PDUs, configured for the selected connection, can be found This reference is used to validate/generate PDUs information for the assigned PDUs for the selected connection over CAN-FD. Dependency on parameter(s): ► XcpR connection type (`XcpRDestinationInterfaceType`): The selected value for this parameter has to be `XcpRDestinationOverCANFD`. Otherwise this parameter is not used.

	► A valid CanIf configuration has to be avaible for selection.
Multiplicity	1..1
Type	REFERENCE
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

5.3.1.20. XcpRDestinationConnectionOverFlexRay

Parameters included	
Parameter name	**Multiplicity**
XcpRDestinationFlxNodeAddress	1..1
XcpRDestinationFlxHeaderAlignment	1..1
XcpRDestinationPackMultiMsgInOneFlexRayFrame	1..1
XcpRDestinationSequenceCorrectionEnabled	1..1
XcpRDestinationMaxFlexMsgLength	1..1
XcpRDestinationConnectionFrIfCfgRef	1..1

Parameter Name	**XcpRDestinationFlxNodeAddress**
Label	Node Address for XcpR (NAX)
Description	Defines the node address for XcpR (NAX) that is used for this node. This value will be written to the NAX field of the FlexRay XcpR header when a message is sent to the master/slave. When multiple FlexRay are configured, it is allowed to share the same NAX address between connections, or to have different NAX address. Range: ► 0 .. 255
Multiplicity	1..1
Type	INTEGER
Default value	0
Range	<=255 >=0
Configuration class	**VariantPreCompile:** VariantPreCompile

Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpRDestinationFlxHeaderAlignment**
Label	Alignment of the FlexRay XcpR Header
Description	Defines the alignment of the FlexRay XcpR header. Range: ► PACKET_ALIGNMENT_8: 8 bit alignment is used. ► PACKET_ALIGNMENT_16: 16 bit alignment is used. ► PACKET_ALIGNMENT_32: 32 bit alignment is used. **Optimization Effect:** ► **RAM reduction (code):** Selecting a smaller value for this parameter reduces the RAM consumption of the module code.
Multiplicity	1..1
Type	ENUMERATION
Default value	PACKET_ALIGNMENT_8
Range	PACKET_ALIGNMENT_8 PACKET_ALIGNMENT_16 PACKET_ALIGNMENT_32
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpRDestinationPackMultiMsgInOneFlexRayFrame**
Label	Enable Multiple XcpR Msgs In One FlexRay Frame
Description	Enables the concatenation of multiple XcpR messages in one frame, for FlexRay communication
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpRDestinationSequenceCorrectionEnabled**
Label	Enable Sequence Correctness

Description	Enables the sequence correction. Each sent frame will contain a Counter that is incremented for each XcpR packet.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**XcpRDestinationMaxFlexMsgLength**
Label	Max Flexray Msg Length
Description	Defines the maximum data length of a FlexRay frame (*MAX_FLX_LEN_BUF_x_init*) that XcpR is able to transmit in a specific XcpR-dedicated buffer. The length of the message can be variable. However the maximum length of an XcpR on FlexRay message is *MAX_FLX_LEN_ABS* = 254 bytes. *MAX_FLX_LEN_BUF_x* is the actual maximum data length of a FlexRay frame in a specific buffer and which can be non-configurable.Thus making *MAX_FLX_LEN_BUF_x* always equal to *MAX_FLX_LEN_BUF_x_init*. Present implementation uses the non-configurable *MAX_FLX_LEN_BUF_x*. Thus for present implementation *MAX_FLX_LEN_BUF_x* equals *MAX_FLX_LEN_BUF_x_init* for all buffers. Range: ► 8 .. 254 Dependency on other parameter(s): This parameter is available if all of the following conditions are fulfilled: ► Multiple XcpR Msgs In One XcpR FlexRay Frame (`XcpRDestinationPackMultiMsgInOneFlexRayFrame`) is enabled. Hint: this parameter should be high enough to be able to accomodate at least 2 maximum size DTOs
Multiplicity	1..1
Type	INTEGER
Default value	254
Configuration class	**PreCompile:** VariantPreCompile

Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpRDestinationConnectionFrIfCfgRef**	
Label	Reference to FrIf configuration	
Description	Reference to the FrIf interface where the Source/Destination PDUs, configured for the selected connection, can be found This reference is used to validate/generate PDUs information for the assigned PDUs for the selected connection over FlexRay. Dependency on parameter(s): ► XcpR destination interface type (`XcpRDestinationInterfaceType`): The selected value for this parameter has to be `XcpRDestinationOver-FlexRay`. Otherwise this parameter is not used. ► A valid FrIf configuration has to be avaible for selection.	
Multiplicity	1..1	
Type	REFERENCE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

5.3.1.21. XcpRDestinationConnectionOverEthernet

Parameters included	
Parameter name	**Multiplicity**
XcpRDestinationOpenSoCon	1..1
XcpRDestinationPackMultiMsgInOneEthernetFrame	1..1
XcpRDestinationMaxEthernetMsgLength	1..1
XcpRDestinationConnectionSoAdConfigRef	1..1

Parameter Name	**XcpRDestinationOpenSoCon**
Label	Open the Socket Connection
Description	Enables XcpR to handle opening of SoAd sockets(s) during startup. If enabled `SoAd_OpenSoCon()` will be called during `XcpR_Init()` for each configured socket. If UDP protocol is used, this parameter must be enabled.

	Note: Please ensure that `SoAd_Init()` will be called before `XcpR_Init()`. Dependency on parameter(s): ► XcpR connection type (`XcpRDestinationConnectionInterfaceType`): The selected value for this parameter has to be `XcpRDestinationConnectionOverEthernet`. Otherwise this parameter is not used.
Multiplicity	1..1
Type	BOOLEAN
Default value	true
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpRDestinationPackMultiMsgInOneEthernetFrame**
Label	Enable Multiple XcpR Msgs In One Ethernet Frame
Description	Enables the concatenation of multiple XcpR messages in one frame, for Ethernet communication. Note: Caution must be taken when enabling this feature, in case the selected communication type is TCP/IP, because the Master should be able to support the packing of multiple pdus in one frame, for TCP/IP, too. Rationale: In ASAM specifications only for UDP/IP is specified that the "Suport for multiple PDUs in one frame may be implemented"
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpRDestinationMaxEthernetMsgLength**
Label	Max Eternet Msg Length
Description	Defines the maximum data length of a Ethernet frame that a specific slave is able to receive or transmit in a frame. It is measured in bytes The length of the message can be variable. However the maximum length of an XcpR on Eternet message is limited to the Ethernet payload length limits in bytes. Range:

	► 8 .. 1454 Dependency on other parameter(s): This parameter is available only if both of the following parameters are enabled: ► XcpR on Ethernet (`XcpROnEthernetEnabled`). ► Multiple XcpR Msgs In One XcpR Frame (`XcpRPackMultiMsgInOneEthernetFrame`). Default value for this parameter is calculated to always be able to accomodate 1 maximum size CTO plus 5 maximum size DTOs Hint this parameter should be high enough to be able to accomodate at least 2 maximum size DTOs
Multiplicity	1..1
Type	INTEGER
Default value	1454
Configuration class	**PreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpRDestinationConnectionSoAdConfigRef**
Label	Reference to SoAd configuration
Description	Reference to the SoAd interface where the Source/Destination PDUs, configured for the selected connection, can be found This reference is used to validate/generate PDUs information for the assigned PDUs for the selected connection over Ethernet. Dependency on parameter(s): ► XcpR connection type (`XcpRDestinationConnectionInterfaceType`): The selected value for this parameter has to be `XcpRDestinationConnectionOverEthernet`. Otherwise this parameter is not used. ► A valid SoAd configuration has to be avaible for selection. ► One entry with the name 'XcpR', containing the UL configuration for XcpR module in SoAd, has to be added into the `SoAdBswModules` container. ► `SoAdIf` has to be enabled, for the dedicated XcpR configuration entry into `SoAdBswModules`. ► `SoAdTp` has to be disabled, for the dedicated XcpR configuration entry into `SoAdBswModules`

	▶ `SoAdIfTxConfirmation` has to be enabled, for the dedicated XcpR configuration entry into `SoAdBswModules` entry. ▶ `SoAdLocalIpAddrAssigmentChg` has to be disabled, for the dedicated XcpR configuration entry into `SoAdBswModules`(refer to ASAM specifications). ▶ `SoAdIfTriggerTransmit` has to be disabled , for the dedicated XcpR configuration entry into `SoAdBswModules`(the trigger transmit api is not supported for Ethernet). ▶ `SoAdSoConModeChg` parameter, configured for the dedicated XcpR configuration entry from `SoAdBswModules` container, has to be:. ▶ Enabled, if there is configured at least one XcpR connection over TCP/IP. ▶ Disabled, if there is no XcpR connection over TCP/IP, configured.
Multiplicity	1..1
Type	REFERENCE
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

5.3.1.22. XcpRDestinationConnectionOverCDD

Containers included		
Container name	**Multiplicity**	**Description**
XcpRCddInformation	1..1	**Label:** CDD Information

5.3.1.23. XcpRCddInformation

Parameters included	
Parameter name	**Multiplicity**
XcpRCddShortName	1..1
XcpRCddHeaderFile	1..1
XcpRCddTransmitFunctionName	1..1

Parameter Name	**XcpRCddShortName**

Label	Cdd Short Name	
Description	Short Name of the CDD module interacting with XcpR.	
Multiplicity	1..1	
Type	STRING	
Default value	Cdd	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**XcpRCddHeaderFile**	
Label	CDD Header File	
Description	Defines the header file for the CDD interface APIs.	
Multiplicity	1..1	
Type	STRING	
Default value	Cdd.h	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**XcpRCddTransmitFunctionName**	
Label	CDD Transmit Function Name	
Description	XcpRCddTransmitFunctionName defines the name of the <Cdd>_Transmit.	
Multiplicity	1..1	
Type	FUNCTION-NAME	
Default value	Cdd_Transmit	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

5.3.1.24. XcpRPdu

Containers included		
Container name	**Multiplicity**	**Description**
XcpRRxPdu	1..1	Defines the Rx PDUs.
XcpRTxPdu	1..1	Defines the Tx PDU.

5.3.1.25. XcpRRxPdu

Parameters included	
Parameter name	**Multiplicity**
XcpRRxDestinationPduReference	1..1

Parameter Name	XcpRRxDestinationPduReference	
Label	Reference to Rx destination PDU	
Description	Reference to Rx PDu.	
Multiplicity	1..1	
Type	REFERENCE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

5.3.1.26. XcpRTxPdu

Parameters included	
Parameter name	**Multiplicity**
XcpRTxDestinationPduReference	1..1
XcpRTxDestinationPduSupportForCddTriggerTransmit	1..1
XcpRTxDestinationPduSupportForFrTriggerTransmit	1..1
XcpRLowerLayerDestinationPduId	1..1
XcpRTxDestinationPduId	1..1
XcpRDestPduSupportTxFromTxConfirmation	1..1

Parameter Name	XcpRTxDestinationPduReference	
Label	Reference to Tx destination PDU	
Description	Reference to Tx Pdu.	
Multiplicity	1..1	
Type	REFERENCE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	XcpRTxDestinationPduSupportForCddTriggerTransmit

Label	PDU Support for CDD Trigger Transmit	
Description	Enables the Tx PDU to support TriggerTransmit functionality over a CDD connection.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**XcpRTxDestinationPduSupportForFrTriggerTransmit**	
Label	PDU Support for FR Trigger Transmit	
Description	Enables the Tx PDU to support TriggerTransmit functionality over a FR connection.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**XcpRLowerLayerDestinationPduId**	
Label	Lower Layer Destination PDU Id	
Description	Defines the ID of the PDU that will be used for outgoing LL_Transmit calls.	
Multiplicity	1..1	
Type	INTEGER	
Default value	0	
Range	<=65535 >=0	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**XcpRTxDestinationPduId**
Label	XcpRTxDestinationPduId
Description	Defines the PDU identifier, which has to be used by the lower layer BSW module for XcpR_TxConfirmation() or XcpR_TriggerTransmit().

	The XcpR's Tx Pdu ids has to be zero based and consecutive. Range: 0 .. 65535
Multiplicity	1..1
Type	INTEGER
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpRDestPduSupportTxFromTxConfirmation**
Label	Support for transmission from TxConfirmation
Description	Support for transmission from Tx confirmation. If enabled a new transmission for this PDU will be triggered when XcpR_TxConfirmation() is executed. **Enabling this feature for several PDUs might have impact on the run-time by creating a big overhead in the system.** Dependency on parameter(s): none.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

5.3.1.27. XcpRDestUpperLayerInformation

Parameters included	
Parameter name	**Multiplicity**
XcpRUpperLayerModuleShortName	1..1
XcpRUpperLayerHeaderFile	1..1
XcpRRxIndicationFunctionName	1..1
XcpRTxConfirmationFunctionName	1..1
XcpRTriggerTransmitFunctionName	1..1

Parameter Name	**XcpRUpperLayerModuleShortName**
Label	UL Module Short Name

Description	Short Name of the upper layer module interacting with XcpR.	
Multiplicity	1..1	
Type	STRING	
Default value	Xcp	
Configuration class	**PreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**XcpRUpperLayerHeaderFile**	
Label	UL Header File	
Description	Defines the header file for the upper layer module interface APIs.	
Multiplicity	1..1	
Type	STRING	
Default value	XcpOnCdd_Cbk.h	
Configuration class	**PreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**XcpRRxIndicationFunctionName**	
Label	UL Rx Indication Function Name	
Description	XcpRRxIndicationFunctionName defines the name of the <UL>_RxIndication.	
Multiplicity	1..1	
Type	FUNCTION-NAME	
Default value	Xcp_CddRxIndication	
Configuration class	**PreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**XcpRTxConfirmationFunctionName**
Label	UL Tx Confirmation Function Name
Description	XcpRTxConfirmationFunctionName defines the name of the <UL>_TxConfirmation.
Multiplicity	1..1
Type	FUNCTION-NAME
Default value	Xcp_CddTxConfirmation

Configuration class	**PreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**XcpRTriggerTransmitFunctionName**	
Label	UL Trigger Transmit Function Name	
Description	XcpRTriggerTransmitFunctionName defines the name of the <UL>_Trigger-Transmit.	
Multiplicity	1..1	
Type	FUNCTION-NAME	
Default value	Xcp_CddTriggerTransmit	
Configuration class	**PreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

5.3.1.28. XcpRDestinationPDUAttributes

Parameters included	
Parameter name	**Multiplicity**
XcpRDestMaxCto	1..1
XcpRDestMaxDto	1..1
XcpRDestMaxCtoPgm	0..1
XcpRDestTimestampType	1..1

Parameter Name	**XcpRDestMaxCto**
Label	Maximum CTO Length
Description	The maximum length of a CTO packet in bytes, specific to each individual destination connection. A CTO packet consists of: ► An Identification Field (the PID containing the CTO Packet code) ► And the Data Field which contains the specific payload necessary for the different types of CTO packet. The length of the CTO varies according to the needs and the used XCP Transport Layer. Range:

	► CAN : 8 ► CAN-FD : 8..(maximum data length of a CAN-FD frame ► FlexRay : 8..(maximum data length of a FlexRay frame (`XcpRDestinationMaxFlexMsgLength`) - FlexRay header length) ► Ethernet : 8..(maximum data length of a Ethernet frame (`XcpRDestinationMaxEthernetMsgLength`) - Ethernet header length) **Optimization Effect:** ► **RAM reduction (code):** Selecting a smaller value for this parameter reduces the RAM consumption of the module code.
Multiplicity	1..1
Type	INTEGER
Default value	8
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**XcpRDestMaxDto**
Label	Maximum DTO Length
Description	The maximum length of a DTO packet in bytes, specific to each individual destination connection. A DTO packet consists of: ► An Identification Field which varies depending upon the Identification Field Type ► An optional Timestamp Field which varies depending upon the Timestamp Field Type(`XcpRDestTimestampType`) ► And the Data Field which contains the data for synchronous acquisition and stimulation. The length of the CTO varies according to the needs and the used XCP Transport Layer. Range: ► CAN : 8 ► CAN-FD : 8..(maximum data length of a CAN-FD frame ► FlexRay : 8..(maximum data length of a FlexRay frame (`XcpRDestinationMaxFlexMsgLength`) - FlexRay header length)

	► Ethernet : 8..(maximum data length of a Ethernet frame (`XcpRDestinationMaxEthernetMsgLength`) - Ethernet header length) **Optimization Effect:** ► **RAM reduction (code):** Selecting a smaller value for this parameter reduces the RAM consumption of the module code. ► **RAM reduction (config):** Selecting a smaller value for this parameter reduces the RAM consumption of the module config. ► **ROM reduction (config):** Selecting a smaller value for this parameter reduces the ROM consumption of the module config.
Multiplicity	1..1
Type	INTEGER
Default value	8
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	AUTOSAR_ECUC

Parameter Name	**XcpRDestMaxCtoPgm**
Label	XcpRMaxCtoPgm
Description	Defines the maximum length of command transfer objects (CTO) in bytes for flash programming.(ASAM parameter: `MAX_CTO_PGM`) A CTO packet consists of: ► An Identification Field (the PID containing the CTO Packet code) ► And the Data Field which contains the specific payload necessary for the different types of CTO packet. The length of the CTO varies according to the needs and the used XCP Transport Layer. Range: ► CAN : 8 ► CAN-FD : 8..(maximum data length of a CAN-FD frame ► FlexRay : 8..(maximum data length of a FlexRay frame `XcpRDestinationMaxFlexMsgLength` - FlexRay header length) Note: Some older Xcp masters ignore this parameter. If your master is affected, it is advised to configure this parameter to the same value as XcpRDestMaxCto. **Optimization Effect:**

	► **RAM reduction (code):** Selecting a smaller value for this parameter reduces the RAM consumption of the module code.
Multiplicity	0..1
Type	INTEGER
Default value	8
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	AUTOSAR_ECUC

Parameter Name	**XcpRDestTimestampType**
Label	Timestamp Type
Description	Defines the type of timestamp XcpR Destination is using. Range: ► NO_TIME_STAMP: no timestamp is used. ► ONE_BYTE: timestamp on 1 byte is used. ► TWO_BYTE: timestamp on 2 bytes is used. ► FOUR_BYTE: timestamp on 4 bytes is used. This parameter must be set with the same value selected for the XcpTimestampType of the Xcp which is connected through this routing path. Every source that is configured as a path for this destination, must have the same value for the parameter XcpRSrcTimestampType.
Multiplicity	1..1
Type	ENUMERATION
Default value	NO_TIME_STAMP
Range	NO_TIME_STAMP ONE_BYTE TWO_BYTE FOUR_BYTE
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

5.3.1.29. XcpRRoutingPaths

Parameters included	
Parameter name	**Multiplicity**
XcpRSourcePduRef	1..1
XcpRDestinationPduRef	1..1

Parameter Name	**XcpRSourcePduRef**	
Label	XcpR Source PDU Reference	
Description	Reference to an XcpR source.	
Multiplicity	1..1	
Type	REFERENCE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**XcpRDestinationPduRef**	
Label	XcpR Destination PDU Reference	
Description	Reference to an XcpR destination.	
Multiplicity	1..1	
Type	REFERENCE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

5.3.1.30. XcpRRoutingToConnectionMapping

Containers included		
Container name	**Multiplicity**	**Description**
XcpRRoutingMapping	2..n	**Label:** XcpR Connection Mapping Container holding a reference to a configured Routing Paths. Each connection group shall be composed of at least: ► A routing path for requests from a master Xcp to a slave Xcp. ► A routing path for the responses from the slave Xcp to the master Xcp.

Containers included		
		Additional routing paths can be mapped to the group for DAQ/STIM.

5.3.1.31. XcpRRoutingMapping

Parameters included	
Parameter name	**Multiplicity**
XcpRRoutingPathRef	1..1

Parameter Name	XcpRRoutingPathRef	
Label	XcpR Routing Path Reference	
Description	Reference to a configured XcpR Routing Path.	
Multiplicity	1..1	
Type	REFERENCE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

5.3.2. Application programming interface (API)

5.3.2.1. Macro constants

5.3.2.1.1. XCPR_AR_RELEASE_MAJOR_VERSION

Purpose	AUTOSAR release major version.
Value	4U

5.3.2.1.2. XCPR_AR_RELEASE_MINOR_VERSION

Purpose	AUTOSAR release minor version.

Value	0U

5.3.2.1.3. XCPR_AR_RELEASE_REVISION_VERSION

Purpose	AUTOSAR release revision version.
Value	3U

5.3.2.1.4. XCPR_E_EXTERNAL_BUS

Purpose	Runtime Error Code.
Value	0x24U
Description	The error is triggered by XcpR_ProcessActiveDestination() if the transmission on the external Bus cannot be done.

5.3.2.1.5. XCPR_E_INTERNAL_BUFFER_OVERFLOW

Purpose	Runtime Error Code.
Value	0x22U
Description	The error is triggered by XcpR_MainFunction() or XcpR_RxIndication(), when processing an XcpR source, if there is an attempt to write more data in the circular XcpR internal queue, than its currently available size. The initial size is determined by the XcpRQueueSize configuration parameter.

5.3.2.1.6. XCPR_E_INTERNAL_BUS

Purpose	Runtime Error Code.
Value	0x23U
Description	The error is triggered by XcpR_ProcessActiveDestination() if the transmission on the internal Bus cannot be done.

5.3.2.1.7. XCPR_E_INVALID_ACTIVE_DESTINATION

Purpose	Runtime Error Code.

Value	0x30U
Description	The error is triggered by XcpR_ProcessSource() if no configured destination has been successfully linked.

5.3.2.1.8. XCPR_E_INVALID_CONNECT_MODE

Purpose	Error Code.
Value	0x14U
Description	The error is triggered by XcpR_MainFunction() if, for the currently processed source with CONNECT mode 0 there is no linked Rx destination or for a source with CONNECT mode 1 there is no linked Tx destination. transmission on the internal Bus cannot be done.

5.3.2.1.9. XCPR_E_INVALID_LENGTH

Purpose	Error Code.
Value	0x11U
Description	Message does not fit in the destination's available length.

5.3.2.1.10. XCPR_E_INVALID_MESSAGE_LENGTH

Purpose	Development Error Code.
Value	0x28U
Description	The error is triggered by XcpR_MainFunction() or XcpR_RxIndication(), when processing an XcpR source, if the data length of the unpacked message is bigger than the maximum between MaxCto/MaxDto/MaxCtoPgm of all configured XcpR sources.

5.3.2.1.11. XCPR_E_INVALID_PDUID

Purpose	Error Code.
Value	0x10U
Description	API called with wrong PDU ID.

5.3.2.1.12. XCPR_E_INVALID_RX_PDU_LENGTH

Purpose	Error Code.
Value	0x12U
Description	The error is triggered by XcpR_RxIndication() if the length of the received Pdu is greater than maximum configured buffer size for the incoming PDUs.

5.3.2.1.13. XCPR_E_INVALID_TX_PDU_LENGTH

Purpose	Error Code.
Value	0x13U
Description	The error is triggered by XcpR_Transmit() if the length of the transmitted Pdu is greater than maximum configured buffer size for the outgoing PDUs.

5.3.2.1.14. XCPR_E_NOT_INITIALIZED

Purpose	Error Code.
Value	0x01U
Description	API is called before complete initialization.

5.3.2.1.15. XCPR_E_NO_TX_EXTERNAL_BUS

Purpose	Runtime Error Code.
Value	0x26U
Description	The error is triggered by XcpR_ProcessBusMonitor() if the Tx confirmation from Xcp master is not received.

5.3.2.1.16. XCPR_E_NO_TX_INTERNAL_BUS

Purpose	Runtime Error Code.
Value	0x25U
Description	The error is triggered by XcpR_ProcessBusMonitor() if the Tx confirmation from remote Xcp slave is not received.

5.3.2.1.17. XCPR_E_NULL_POINTER

Purpose	Error Code.
Value	0x02U
Description	Null pointer has been passed as an argument.

5.3.2.1.18. XCPR_E_PDU_BUFFER_FULL

Purpose	Runtime Error Code.
Value	0x27U
Description	The error is triggered by XcpR_RxIndication() if the message is lost because the assigned receive Pdu buffer is full.

5.3.2.1.19. XCPR_E_PDU_LOST

Purpose	Runtime Error Code.
Value	0x20U
Description	The error is triggered if XcpR is disconnected and if the PID of the received Pdu is not a CONNECT PID or if the response after connect/disconnect is not positive.

5.3.2.1.20. XCPR_E_PDU_OUTSIDE_CONNECTION_GROUP

Purpose	Development Error Code.
Value	0x29U
Description	The error is triggered by XcpR_RxIndication() if the PduId is not accepted because is not part of the currently active Connection Group, which is set after XcpR is in state CONNECTED, considering the configuration.

5.3.2.1.21. XCPR_E_UNEXPECTED_MSG

Purpose	Runtime Error Code.
Value	0x21U

Description	The error is triggered by XcpR_MainFunction() when processing a source or a destination, if the XcpR is disconnected, and XcpR receives a response from the local or remote Xcp slave but the previous command was not a CONNECT command.

5.3.2.1.22. XCPR_INSTANCE_ID

Purpose	Id of instance of XcpR.
Value	0U

5.3.2.1.23. XCPR_MODULE_ID

Purpose	AUTOSAR module identification.
Value	255U

5.3.2.1.24. XCPR_SID_GET_VERSION_INFO

Purpose	API service ID.
Value	0x01U
Description	Definition of service ID for XcpR_GetVersionInfo().

5.3.2.1.25. XCPR_SID_IF_RX_INDICATION

Purpose	API service ID.
Value	0x03U
Description	Definition of the general service ID for XcpR_RxIndication() function.

5.3.2.1.26. XCPR_SID_IF_TRANSMIT

Purpose	API service ID.
Value	0x04U
Description	Definition of service ID for XcpR_Transmit().

5.3.2.1.27. XCPR_SID_IF_TRIGGER_TRANSMIT

Purpose	API service ID.
Value	0x05U
Description	Definition of service ID for XcpR_TriggerTransmit().

5.3.2.1.28. XCPR_SID_IF_TX_CONFIRMATION

Purpose	API service ID.
Value	0x02U
Description	Definition of the general service ID for XcpR_TxConfirmation() function.

5.3.2.1.29. XCPR_SID_INIT

Purpose	API service ID.
Value	0x00U
Description	Definition of service ID for XcpR_Init().

5.3.2.1.30. XCPR_SID_MAIN_FUNCTION

Purpose	API service ID.
Value	0x06U
Description	Definition of service ID for XcpR_MainFunction().

5.3.2.1.31. XCPR_SID_SOAD_SO_CON_MODE_CHG

Purpose	API service ID.
Value	0x07U
Description	Definition of service ID for XcpR_SoAdSoConModeChg().

5.3.2.1.32. XCPR_SW_MAJOR_VERSION

Purpose	AUTOSAR module major version.

Value	1U

5.3.2.1.33. XCPR_SW_MINOR_VERSION

Purpose	AUTOSAR module minor version.
Value	1U

5.3.2.1.34. XCPR_SW_PATCH_VERSION

Purpose	AUTOSAR module patch version.
Value	1U

5.3.2.1.35. XCPR_VENDOR_ID

Purpose	AUTOSAR vendor identification: Elektrobit Automotive GmbH.
Value	1U

5.3.2.2. Functions

5.3.2.2.1. XcpR_GetVersionInfo

Purpose	Return the modules version information.	
Synopsis	`void XcpR_GetVersionInfo ( Std_VersionInfoType * VersionInfoPtr );`	
Service ID	XCPR_SID_GET_VERSION_INFO	
Sync/Async	Synchronous	
Reentrancy	Non-Reentrant	
Parameters (out)	`VersionInfoPtr`	Pointer to where to store the version information of this module.
Description	This function provides the information to module vendor ID, module ID and software version major.minor.patch Precondition: XCPR_VERSION_INFO_API = STD_ON	

5.3.2.2.2. XcpR_Init

Purpose	Initializes the XCPR.
Synopsis	`void` **`XcpR_Init`** `( void );`
Service ID	XCPR_SID_INIT
Sync/Async	Synchronous
Reentrancy	Non-Reentrant
Description	This function initializes interfaces and variables of the XCP Router module. Precondition: None.

5.3.2.2.3. XcpR_MainFunction

Purpose	Scheduled function of the XCP Router module.
Synopsis	`void` **`XcpR_MainFunction`** `( void );`
Service ID	XCPR_SID_MAIN_FUNCTION
Sync/Async	Synchronous
Reentrancy	Non-Reentrant
Description	Scheduled function of the XCP Router module. Precondition: None.

5.3.2.2.4. XcpR_RxIndication

Purpose	XCPR PDU is received.	
Synopsis	`void` **`XcpR_RxIndication`** `( PduIdType XcpRRxSrcPduId , PduInfoType * XcpRRxSrcPduPtr );`	
Parameters (in)	`XcpRRxSrcPduId`	PDU-ID that has been received.
Parameters (out)	`XcpRRxSrcPduPtr`	Pointer to SDU (Buffer of received payload).
Description	This function is called by the lower layer module when an XCPR PDU has been received.	

5.3.2.2.5. XcpR_SoAdSoConModeChg

Purpose	Socket connection mode change notification.	
Synopsis	`void` **`XcpR_SoAdSoConModeChg`** `( SoAd_SoConIdType SoConId , SoAd_SoConModeType Mode );`	
Service ID	XCPR_SID_SOAD_SO_CON_MODE_CHG	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different SoConIds. Non reentrant for the same SoConId	
Parameters (in)	`SoConId`	Socket connection ID of socket that has changed
	`Mode`	Socket connection mode (ONLINE, OFFLINE or RECONNECT)
Description	This function is called by the lower layer Socket Adapter when the socket connection mode has changed. If the socket is closed while in XcpR connected state, the XcpR module will perform an autonomous XcpR disconnect, which means that all communication will be stopped. This function is available only for the TCP protocol (not available for UDP because it is a broadcasting protocol)	

5.3.2.2.6. XcpR_TriggerTransmit

Purpose	Call when an XCPR PDU shall be transmitted.	
Synopsis	`Std_ReturnType` **`XcpR_TriggerTransmit`** `( PduIdType XcpRTxPduId , PduInfoType * PduInfoPtr );`	
Parameters (in)	`XcpRTxPduId`	PDU-ID that has been requested to be transmitted.
Parameters (out)	`PduInfoPtr`	Pointer to PduInfo SDU (pointer to SDU buffer and SDU length).
Return Value		
Description	This function is called by the lower layer module when an XCPR PDU shall be transmitted. The function XcpR_TriggerTransmit is called by the lower layer module for requesting the I-PDU before transmission. Whether the function XcpR_TriggerTransmit is called or not is statically configured for each I-PDU in the configuration.	

5.3.2.2.7. XcpR_TxConfirmation

Purpose	XCPR PDU transmit confirmation.	
Synopsis	`void` **`XcpR_TxConfirmation`** `( PduIdType XcpRTxPduId );`	
Parameters (in)	`XcpRTxPduId`	PDU-ID that has been transmitted.
Description	This function is called by the lower layer module when an XCPR PDU has been transmitted.	

5.3.3. Integration notes

5.3.3.1. Exclusive areas

Exclusive areas information is not available for this module.

5.3.3.2. Production errors

Production errors information is not available for this module.

5.3.3.3. Memory mapping

General information about memory mapping is provided in the EB tresos AutoCore Generic documentation. Refer to the section `Memory mapping and compiler abstraction` in the `Integration notes` section for details.

The following table provides the list of sections that may be mapped for this module:

Memory section
CONST_UNSPECIFIED
VAR_8
VAR_CLEARED_8
VAR_UNSPECIFIED
VAR_CLEARED_UNSPECIFIED
CODE

WRAPPER_CODE

5.3.3.4. Integration requirements

WARNING 	**Integration requirements list is not exhaustive** The following list of integration requirements helps you to integrate your product. However, this list is not exhaustive. You also require information from the user's guide, release notes, and EB tresos AutoCore known issues to successfully integrate your product.

5.3.3.4.1. intgr.XcpR.EB_INTREQ_XcpR_0001

Description	The Routing Paths for requests from the Xcp master to the Xcp slave and for responses from Xcp slave to Xcp master, shall be unique between configured Connection Groups.
Rationale	Each configured Connection Group is composed of at least one Routing Path for requests from Xcp master to Xcp slave and for responses from Xcp slave to Xcp master. After XcpR is in state CONNECTED, the Active Connection Group is saved based on these two Routing Paths. When a XcpR_RxIndication() request is made, it is accepted only if it is part of the Active Connection Group. If these two Routing Paths are not unique for each Connection Group, XcpR cannot determine which Connection Group shall be used for forwarding messages.

5.3.3.4.2. intgr.XcpR.EB_INTREQ_XcpR_0002

Description	The master Xcp shall have MaxCto set as the minimum between MaxCto from the local Xcp slave and remote Xcp slave, otherwise messages won't be correctly received.

5.3.3.4.3. intgr.XcpR.EB_INTREQ_XcpR_0003

Description	If "packing of multiple messages in one frame" is enabled on a remote Xcp instance, both XcpR Source and Destination connected to that particular Xcp instance shall have "packing of multiple messages in one frame" enabled.
Rationale	Xcp supports the "packing of multiple messages in one frame" feature. XcpR also supports this feature, but it can be enabled either for a Source, either for a Destination.

5.3.3.4.4. intgr.XcpR.EB_INTREQ_XcpR_0004

Description	If XcpR supports "packing of multiple messages in one frame" for the connection to Master Xcp but not for the connection to the slave Xcp, the XcpR Source from slave Xcp shall have the "receive from rx indication" option enabled and the Master Xcp shall have "packing of multiple messages in one frame" enabled.
Rationale	If one frame with multiple messages received from master Xcp, and the slave Xcp replies with multiple responses, if "receive from rx indication" option is not enabled, the second response will be sent with the same PduId, and the buffer for this PduId is locked in XcpR.

5.3.3.4.5. intgr.XcpR.EB_INTREQ_XcpR_0005

Description	The bus timeout for TxConfirmation in Xcp shall be configured with a higher value than the one in XcpR.
Rationale	XcpR will forward the messages from Xcp and will wait for TxConfirmation which it forwards to Xcp, so Xcp should not disconnect before XcpR.

5.3.3.4.6. intgr.XcpR.EB_INTREQ_XcpR_0006

Description	If FlexRay sequence correction is enabled for a source, then it shall be enabled for its linked remote Xcp destination and also for the Master Xcp.

5.3.3.4.7. intgr.XcpR.EB_INTREQ_XcpR_0007

Description	For connections over CDD, the Upper Layer short name and Upper Layer header file, shall have UpperLayer Cdd Short name and UpperLayer Cdd Header file need to have the same values in all configured Rx Destinations.
Rationale	Examples for connection over CDD with a upper layer Xcp module are: Xcp and XcpOnCdd_Cbk.h

5.3.3.4.8. intgr.XcpR.EB_INTREQ_XcpR_0008

Description	If multiple CDD destinations are configured, then all these destinations shall use the same callback functions.

5.3.3.4.9. intgr.XcpR.EB_INTREQ_XcpR_0009

Description	In the XcpR routing paths list, one source shall be linked to either one destination, or a maximum of 2 destinations: one Rx destination and one Tx destination. A source cannot be linked to 2 Rx destinations or 2 Tx destinations.

5.3.3.4.10. intgr.XcpR.EB_INTREQ_XcpR_0010

Description	Connections between XcpR and a master Xcp, or between XcpR and a slave Xcp, shall have the same bus-type attributes and PDU attributes.
Rationale	For XcpR to be connected, for example, to a master Xcp on FlexRay, it needs to configure a XcpR source on Fr and a XcpR destination on Fr. This source shall have the same characteristics as the destination (i.e. Multiple packing enabled, package alignment, MaxCto etc.)