Document Title	Specification of Ethernet Interface
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	417

Document Status	published
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	R25-11

Document Change History			
Date	**Release**	**Changed by**	**Description**
2025-11-27	R25-11	AUTOSAR Release Management	• Added new requirements and interfaces for MACsec enhancements • Added abstraction support from Driver APIs • Added context data and trigger conditions for the security events • Updated rateDeviation range in EthIf_SetPhcCorrection • Added MKA APIs to Optional Interfaces • Set EthIf_GetIngressTimeStamp to Obsolete • Added new DET error ETHIF_E_INV_CLKUNIT_IDX for invalid ClkUnitIdx • Added optional references to physical controller for an Rx/Tx PDU pool • Created new chapter for Configuration Constraints • Improved VLAN priority usage for transmission • Added new requirements for Ethernet Switch Management • Removed chapters: ▽

▽

△

			– EthIf_GetCurrentTime – EthIf_MainFunctionRx_ <PriorityProcessing ShortName> • Editorial changes
2024-11-27	R24-11	AUTOSAR Release Management	• IEEE1722 streams support • Independent VLAN learning support • Editorial changes
2023-11-23	R23-11	AUTOSAR Release Management	• New chapters for: – Firewall support – Communication • Editorial changes
2022-11-24	R22-11	AUTOSAR Release Management	• Migration from word document to LaTex • Changed [SWS_EthIf_00498] and [SWS_EthIf_00504] to Valid • New chapters for: – CellularV2x (V2X for China) – CAN-XL – MACSec • Editorial changes
2021-11-25	R21-11	AUTOSAR Release Management	• Updates on 10BASE-T1S • EthenetWakeOnDataLine Specification items valid (no “Draft” tag) • Clarification on Return codes and error reporting • Updates on ReworkofPNCrelatedCom-MandNMhandling Concept • Clarification on “Synchronous/Asynchronous” APIs • Removed section EthIfSwitchTimeStampIndicationConfig • Removed [SWS_EthIf_00248] • Update uptraces of Security Event tables

▽

△

2020-11-30	R20-11	AUTOSAR Release Management	• Introduction of Security Event reporting (DRAFT) • Introduction of EthernetWakeupOnDataline (DRAFT) • Introduction of 10BASE-T1S (DRAFT)
2019-11-28	R19-11	AUTOSAR Release Management	• Some empty pages removed • API Table for EthIf_MainFunctionRx_ <PriorityProcessing ShortName> corrected • EthSwt_PortModeType introduced to explicitly distinguish between Port Mode and Transceiver Mode • Missing and duplicate service IDs corrected • Missing API of EthSwt and EthTrcv are added in EthIf • “BSWDistribution” CONC_643 added as draft • Changed Document Status from Final to published
2018-10-31	4.4.0	AUTOSAR Release Management	• Explicite link control in Ethernet transceiver • minor corrections / clarifications / editorial changes; For details please refer to the ChangeDocumentation
2017-12-08	4.3.1	AUTOSAR Release Management	• Improved transceiver tests (Signal quality) • Enhanced sequence charts (Ethernet switch handling)
2016-11-30	4.3.0	AUTOSAR Release Management	• Diagnostics access APIs added • gPTP Timestamp rework • Ethernet Switch enhancements (Port Groups) • Wireless Ethernet support

▽

△

2015-07-31	4.2.2	AUTOSAR Release Management	• EthIf_TransceiverInit and EthIf_ControllerInit removed • Development Error Tracer renamed to Default Error Tracer
2014-10-31	4.2.1	AUTOSAR Release Management	• Change from Synchronous to Asynchronous API • gPTP Timestamp Support • Ethernet Switch Support • Ethernet Wakeup Support
2014-03-31	4.1.3	AUTOSAR Release Management	• Extended UL_RxIndication • Editorial changes
2013-10-31	4.1.2	AUTOSAR Release Management	• Introduction of Eth_GeneralTypes.h • Support of API deviation for asynchronous implementation • Changes in API of EthIf_ProvideTxBuffer and EthIf_SetPhysAddr • Editorial changes • Removed chapter(s) on change documentation
2013-03-15	4.1.1	AUTOSAR Administration	• Remove “Comercial Off The Shelf” use case • VLAN support • 1000MBit Ethernet support
2011-12-22	4.0.3	AUTOSAR Administration	• Description of payload data in EthIf_Cbk_RxIndication adapted
2010-09-30	3.1.5	AUTOSAR Adiministration	• Further post-build configurable parameters • EthIf_MainFunctionTx functional requirements improved (functionality split) • ’Instance ID’ removed from Version Info (concerns EthIf_GetVersionInfo API) • Additional development error in EthIf_GetVersionInfo API

▽

△

2010-02-02	3.1.4	AUTOSAR Administration	• Initial Release

Disclaimer

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

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

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

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

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

Table of Contents

1 Introduction and functional overview 15
2 Acronyms and Abbreviations 17
3 Related documentation 18
3.1 Input documents & related standards and norms . . . 18
3.2 Related specification . . . 19
4 Constraints and assumptions 20
4.1 Limitations . . . 20
4.2 Applicability to car domains . . . 20
5 Dependencies to other modules 21
6 Requirements Tracing 22
7 Functional specification 25
7.1 Ethernet BSW stack . . . 25
7.1.1 Indexing scheme for Ethernet controller . . . 25
7.1.2 Indexing scheme for Ethernet switches . . . 26
7.1.3 Initialization . . . 26
7.1.4 Ethernet Interface main function . . . 27
7.1.5 Requirements . . . 27
7.1.6 Configuration description . . . 28
7.1.7 VLAN support . . . 28
7.1.8 Wake up support . . . 29
7.1.9 Ethernet Switch Management support . . . 29
7.1.10 Handling of maintained Ethernet hardware . . . 31
7.1.10.1 EthIfSwitchPortGroup handling . . . 32
7.1.10.2 Switching of EthIfController and the corresponding Ethernet hardware . . . 32
7.1.10.3 Link state handling . . . 38
7.1.10.4 Additional Ethernet switch port handling . . . 44
7.1.11 Communication control . . . 45
7.1.12 Communication . . . 46
7.1.12.1 Reception . . . 47
7.1.12.2 Transmission . . . 51
7.1.12.3 Transmission confirmation . . . 58
7.1.12.4 State handling of PDUs . . . 58
7.1.12.5 Meta data handling . . . 59
7.1.12.6 Ingress queue handling . . . 61
7.1.13 Global Time support . . . 62
7.1.14 Wireless Ethernet Support . . . 62
7.1.15 Cellular V2X Support . . . 63

7.1.16 MACsec support . . . 64
7.1.16.1 MACsec support in software . . . 74
7.1.17 Firewall support . . . 78
7.2 Security Events . . . 79
7.3 Error Classification . . . 82
7.3.1 Development Errors . . . 82
7.3.2 Runtime Errors . . . 83
7.3.3 Production Errors . . . 83
7.3.4 Extended Production Errors . . . 83
8 API specification 84
8.1 API Parameter Checking . . . 84
8.2 Imported types . . . 85
8.3 Type definitions . . . 87
8.3.1 EthIf_ConfigType . . . 87
8.3.2 EthIf_SwitchPortGroupIdxType . . . 87
8.3.3 EthIf_MeasurementIdxType . . . 88
8.3.4 EthIf_SignalQualityResultType . . . 88
8.4 Function definitions . . . 88
8.4.1 Driver . . . 89
8.4.1.1 EthIf_SetControllerMode . . . 89
8.4.1.2 EthIf_GetControllerMode . . . 89
8.4.1.3 EthIf_ReadMii . . . 90
8.4.1.4 EthIf_WriteMii . . . 91
8.4.1.5 EthIf_ReadMmd . . . 91
8.4.1.6 EthIf_WriteMmd . . . 92
8.4.1.7 EthIf_GetPhysAddr . . . 93
8.4.1.8 EthIf_SetPhysAddr . . . 93
8.4.1.9 EthIf_UpdatePhysAddrFilter . . . 94
8.4.1.10 EthIf_GetPortMacAddr . . . 94
8.4.1.11 EthIf_GetPortMacAddrVlan . . . 95
8.4.1.12 EthIf_GetArlTable . . . 96
8.4.1.13 EthIf_Transmit . . . 97
8.4.1.14 EthIf_EthGetSpiStatus . . . 98
8.4.2 Firewall . . . 98
8.4.2.1 EthIf_GetStreamStatistics . . . 98
8.4.2.2 EthIf_SetStreamState . . . 99
8.4.3 General . . . 100
8.4.3.1 EthIf_Init . . . 100
8.4.3.2 EthIf_GetCtrlIdxList . . . 100
8.4.3.3 EthIf_GetVlanId . . . 101
8.4.3.4 EthIf_GetAndResetMeasurementData . . . 102
8.4.3.5 EthIf_VerifyConfig . . . 103
8.4.3.6 EthIf_SetForwardingMode . . . 103

8.4.3.7 EthIf_ClearTrcvSignalQuality . . . 104
8.4.3.8 EthIf_ClearSwitchPortSignalQuality . . . 104
8.4.3.9 EthIf_ReleaseRxBuffer . . . 105
8.4.3.10 EthIf_GetVersionInfo . . . 106
8.4.4 MacSec . . . 106
8.4.4.1 EthIf_SwitchMacSecUpdateSecY . . . 106
8.4.4.2 EthIf_MacSecUpdateSecY . . . 107
8.4.4.3 EthIf_SwitchMacSecInitRxSc . . . 108
8.4.4.4 EthIf_MacSecInitRxSc . . . 108
8.4.4.5 EthIf_SwitchMacSecResetRxSc . . . 109
8.4.4.6 EthIf_MacSecResetRxSc . . . 110
8.4.4.7 EthIf_SwitchMacSecAddTxSa . . . 110
8.4.4.8 EthIf_MacSecAddTxSa . . . 111
8.4.4.9 EthIf_SwitchMacSecUpdateTxSa . . . 112
8.4.4.10 EthIf_MacSecUpdateTxSa . . . 113
8.4.4.11 EthIf_SwitchMacSecDeleteTxSa . . . 113
8.4.4.12 EthIf_MacSecDeleteTxSa . . . 114
8.4.4.13 EthIf_SwitchMacSecAddRxSa . . . 115
8.4.4.14 EthIf_MacSecAddRxSa . . . 116
8.4.4.15 EthIf_SwitchMacSecUpdateRxSa . . . 117
8.4.4.16 EthIf_MacSecUpdateRxSa . . . 117
8.4.4.17 EthIf_SwitchMacSecDeleteRxSa . . . 118
8.4.4.18 EthIf_MacSecDeleteRxSa . . . 119
8.4.4.19 EthIf_SwitchMacSecGetTxSaNextPn . . . 119
8.4.4.20 EthIf_MacSecGetTxSaNextPn . . . 120
8.4.4.21 EthIf_SwitchMacSecGetMacSecStatistics . . . 121
8.4.4.22 EthIf_MacSecGetMacSecStatistics . . . 121
8.4.4.23 EthIf_SwitchMacSecOperational . . . 122
8.4.4.24 EthIf_MacSecOperational . . . 123
8.4.4.25 EthIf_SwitchMacSecSetControlledPortEnabled . . . 123
8.4.4.26 EthIf_MacSecSetControlledPortEnabled . . . 124
8.4.5 Switch driver . . . 125
8.4.5.1 EthIf_GetSwitchPortMode . . . 125
8.4.5.2 EthIf_GetSwitchPortWakeupReason . . . 125
8.4.5.3 EthIf_StoreConfiguration . . . 126
8.4.5.4 EthIf_ResetConfiguration . . . 127
8.4.5.5 EthIf_SwitchPortGroupRequestMode . . . 127
8.4.5.6 EthIf_StartAllPorts . . . 129
8.4.5.7 EthIf_SetSwitchMgmtInfo . . . 129
8.4.5.8 EthIf_GetRxMgmtObject . . . 130
8.4.5.9 EthIf_GetTxMgmtObject . . . 130
8.4.5.10 EthIf_SwtEthRxProcessFrame . . . 131
8.4.5.11 EthIf_SwtEthRxFinishedIndication . . . 131

8.4.5.12 EthIf_SwtEthTxPrepareFrame . . . 132
8.4.5.13 EthIf_SwtEthTxAdaptBufferLength . . . 133
8.4.5.14 EthIf_SwtEthTxProcessFrame . . . 133
8.4.5.15 EthIf_SwtEthTxFinishedIndication . . . 134
8.4.5.16 EthIf_GetSwitchPortSignalQuality . . . 134
8.4.5.17 EthIf_SwitchPortGetLinkState . . . 135
8.4.5.18 EthIf_SwitchPortGetBaudRate . . . 136
8.4.5.19 EthIf_SwitchPortGetDuplexMode . . . 136
8.4.5.20 EthIf_SwitchPortReadTrcvRegister . . . 137
8.4.5.21 EthIf_SwitchPortWriteTrcvRegister . . . 138
8.4.5.22 EthIf_SwitchPortReadMmd . . . 138
8.4.5.23 EthIf_SwitchPortWriteMmd . . . 139
8.4.5.24 EthIf_SwitchPortGetCounterValues . . . 140
8.4.5.25 EthIf_SwitchPortGetRxStats . . . 140
8.4.5.26 EthIf_SwitchPortGetTxStats . . . 141
8.4.5.27 EthIf_SwitchPortGetTxErrorCounterValues . . . 142
8.4.5.28 EthIf_SwitchPortGetMacLearningMode . . . 142
8.4.5.29 EthIf_GetSwitchPortIdentifier . . . 143
8.4.5.30 EthIf_GetSwitchIdentifier . . . 144
8.4.5.31 EthIf_WritePortMirrorConfiguration . . . 144
8.4.5.32 EthIf_ReadPortMirrorConfiguration . . . 145
8.4.5.33 EthIf_DeletePortMirrorConfiguration . . . 146
8.4.5.34 EthIf_GetPortMirrorState . . . 146
8.4.5.35 EthIf_SetPortMirrorState . . . 147
8.4.5.36 EthIf_SetPortTestMode . . . 148
8.4.5.37 EthIf_SetPortLoopbackMode . . . 148
8.4.5.38 EthIf_SetPortTxMode . . . 149
8.4.5.39 EthIf_GetPortCableDiagnosticsResult . . . 150
8.4.5.40 EthIf_RunPortCableDiagnostic . . . 150
8.4.5.41 EthIf_SwitchGetCfgDataRaw . . . 151
8.4.5.42 EthIf_SwitchGetCfgDataInfo . . . 152
8.4.5.43 EthIf_SwitchPortGetMaxQueueBufferFillLevel . . . 152
8.4.6 TimeSync . . . 153
8.4.6.1 EthIf_SwitchEnableTimeStamping . . . 153
8.4.6.2 EthIf_GetCurrentTimeTuple . . . 154
8.4.6.3 EthIf_EnableEgressTimeStamp . . . 155
8.4.6.4 EthIf_GetEgressTimeStamp . . . 155
8.4.6.5 EthIf_GetIngressTimeStamp . . . 156
8.4.6.6 EthIf_SetPhcTime . . . 157
8.4.6.7 EthIf_SetPhcCorrection . . . 158
8.4.6.8 EthIf_GetPhcTime . . . 159
8.4.6.9 EthIf_SetPpsSignalMode . . . 160
8.4.7 Transceiver Driver . . . 161

8.4.7.1 EthIf_CheckWakeup . . . 161
8.4.7.2 EthIf_GetPhyWakeupReason . . . 162
8.4.7.3 EthIf_GetTrcvSignalQuality . . . 163
8.4.7.4 EthIf_SetPhyTestMode . . . 164
8.4.7.5 EthIf_SetPhyLoopbackMode . . . 164
8.4.7.6 EthIf_SetPhyTxMode . . . 165
8.4.7.7 EthIf_GetCableDiagnosticsResult . . . 166
8.4.7.8 EthIf_GetPhyIdentifier . . . 166
8.4.7.9 EthIf_GetTransceiverMode . . . 167
8.4.7.10 EthIf_SetTransceiverMode . . . 168
8.4.7.11 EthIf_StartAutoNegotiation . . . 168
8.4.7.12 EthIf_TransceiverGetLinkState . . . 169
8.4.7.13 EthIf_TransceiverGetBaudRate . . . 169
8.4.7.14 EthIf_TransceiverGetDuplexMode . . . 170
8.4.7.15 EthIf_RunCableDiagnostic . . . 171
8.4.7.16 EthIf_TransceiverGetMacMethod . . . 171
8.4.8 Wireless(CC2x) . . . 172
8.4.8.1 EthIf_GetBufWRxParams . . . 172
8.4.8.2 EthIf_GetBufWTxParams . . . 173
8.4.8.3 EthIf_SetBufWTxParams . . . 173
8.4.8.4 EthIf_SetRadioParams . . . 174
8.4.8.5 EthIf_SetChanRxParams . . . 175
8.4.8.6 EthIf_SetChanTxParams . . . 176
8.4.8.7 EthIf_GetChanRxParams . . . 176
8.4.8.8 EthIf_GetBufCV2xPC5RxParams . . . 177
8.4.8.9 EthIf_GetBufCV2xPC5TxParams . . . 178
8.4.8.10 EthIf_SetBufCV2xPC5TxParams . . . 179
8.4.8.11 EthIf_GetChanCV2xPC5TxParams . . . 180
8.5 Callback notifications . . . 180
8.5.1 EthIf_RxIndication . . . 181
8.5.2 EthIf_TxConfirmation . . . 182
8.5.3 EthIf_CtrlModeIndication . . . 182
8.5.4 EthIf_TrcvModeIndication . . . 183
8.5.5 EthIf_SwitchPortModeIndication . . . 183
8.5.6 EthIf_SleepIndication . . . 184
8.5.7 EthIf_StreamStateIndication . . . 185
8.5.8 EthIf_StreamStatisticsIndication . . . 185
8.5.9 EthIf_SwitchMacSecGetMacSecStatisticsNotification . . . 186
8.5.10 EthIf_MacSecGetMacSecStatisticsNotification . . . 187
8.5.11 EthIf_SwitchMacSecUpdateSecYNotification . . . 187
8.5.12 EthIf_MacSecUpdateSecYNotification . . . 188
8.5.13 EthIf_SwitchMacSecAddTxSaNotification . . . 189
8.5.14 EthIf_MacSecAddTxSaNotification . . . 189

8.5.15 EthIf_SwitchMacSecAddRxSaNotification . . . 190
8.5.16 EthIf_MacSecAddRxSaNotification . . . 191
8.5.17 EthIf_EthMacSecUpdateSecYNotification . . . 191
8.5.18 EthIf_EthMacSecAddRxSaNotification . . . 192
8.5.19 EthIf_EthMacSecAddTxSaNotification . . . 193
8.5.20 EthIf_EthMacSecGetMacSecStatisticsNotification . . . 193
8.6 Scheduled functions . . . 194
8.6.1 EthIf_MainFunctionRx . . . 194
8.6.2 EthIf_MainFunctionRx_<IngressQueueProcessing ShortName> . . . 194
8.6.3 EthIf_MainFunctionTx . . . 195
8.6.4 EthIf_MainFunctionState . . . 196
8.7 Expected interfaces . . . 197
8.7.1 Mandatory interfaces . . . 198
8.7.2 Optional interfaces . . . 198
8.7.3 Configurable interfaces . . . 205
9 Sequence diagrams 207
9.1 Initialization . . . 207
9.2 Communication Initialization . . . 208
9.3 Switch Initialization . . . 208
9.4 Data Transmission . . . 209
9.5 Data Reception . . . 210
9.6 Link State Change . . . 211
9.7 Link State Change without Port Groups . . . 211
9.8 Link State Change with Port Groups . . . 212
9.9 Link State Change with Port Groups and Partial Network Cluster . . . 213
9.10 Switch Management support . . . 214
10 Configuration specification 217
10.1 How to read this chapter . . . 217
10.2 Containers and configuration parameters . . . 217
10.2.1 EthIf . . . 225
10.2.2 EthIfGeneral . . . 226
10.2.3 EthIfConfigSet . . . 239
10.2.4 EthIfController . . . 240
10.2.5 EthIfFrameConfig . . . 245
10.2.5.1 EthIfFrameRxPool . . . 246
10.2.5.2 EthIfFrameRxPdu . . . 248
10.2.5.3 EthIfFrameTxPool . . . 249
10.2.5.4 EthIfFrameTxPdu . . . 252
10.2.6 EthIfPhysController . . . 253
10.2.7 EthIfPhysCtrlRxMainFunctionIngressQueueProcessing . . . 256
10.2.8 EthIfClkUnit . . . 258
10.2.9 EthIfRxIndicationConfig . . . 260

10.2.10 EthIfSwitch . . . 260
10.2.11 EthIfSwitchPortGroup . . . 262
10.2.12 EthIfTransceiver . . . 263
10.2.13 EthIfTrcvLinkStateChgConfig . . . 266
10.2.14 EthIfTxConfirmationConfig . . . 266
10.2.15 EthIfSecurityEventRefs . . . 267
10.3 Configuration constraints . . . 270
10.4 Published Information . . . 271
A Not applicable requirements 272
B Change History 273
B.1 Change History of this document according to AUTOSAR Release R25-11 273
B.1.1 Added Constraints in R25-11 . . . 273
B.1.2 Changed Constraints in R25-11 . . . 274
B.1.3 Deleted Constraints in R25-11 . . . 274
B.1.4 Added Specification Items in R25-11 . . . 274
B.1.5 Changed Specification Items in R25-11 . . . 276
B.1.6 Deleted Specification Items in R25-11 . . . 278
B.2 Change History of this document according to AUTOSAR Release R24-11 279
B.2.1 Added Constraints in R24-11 . . . 279
B.2.2 Changed Constraints in R24-11 . . . 279
B.2.3 Deleted Constraints in R24-11 . . . 280
B.2.4 Added Specification Items in R24-11 . . . 280
B.2.5 Changed Specification Items in R24-11 . . . 282
B.2.6 Deleted Specification Items in R24-11 . . . 283
B.3 Change History of this document according to AUTOSAR Release R23-11 289
B.3.1 Added Specification Items in R23-11 . . . 289
B.3.2 Changed Specification Items in R23-11 . . . 291
B.3.3 Deleted Specification Items in R23-11 . . . 291
B.3.4 Added Constraints in R23-11 . . . 292
B.3.5 Changed Constraints in R23-11 . . . 292
B.3.6 Deleted Constraints in R23-11 . . . 292
B.4 Change History of this document according to AUTOSAR Release R22-11 292
B.4.1 Added Specification Items in R22-11 . . . 292
B.4.2 Changed Specification Items in R22-11 . . . 295
B.4.3 Deleted Specification Items in R22-11 . . . 298
B.4.4 Added Constraints in R22-11 . . . 298
B.4.5 Changed Constraints in R22-11 . . . 298
B.4.6 Deleted Constraints in R22-11 . . . 298

Known Limitations

Currently, chapter 5 Dependencies to other modules does not describe the versions of dependent modules. Thus, a version check will extend the chapter.

1 Introduction and functional overview

This specification specifies the functionality, API and the configuration of the AUTOSAR Basic Software module Ethernet Interface.

In the AUTOSAR Layered Software Architecture [1], the Ethernet Interface belongs to the ECU Abstraction Layer, or more precisely, to the Communication Hardware Abstraction.

This indicates the main task of the Ethernet Interface:

Provide to upper layers a hardware independent interface to the Ethernet Communication System comprising multiple different wired or wireless Ethernet controllers and transceivers. This interface shall be uniform for all Ethernet controllers and transceivers, as well as Cellular V2X controllers. Thus, the upper layers (TCP/IP [2], EthSM [3], CDD, V2x modules) may access the underlying bus system in a uniform manner.

The Ethernet Interface does not directly access the Ethernet hardware (Ethernet Communication Controller and Ethernet Transceiver) but by means of one or more hardware-specific driver modules.

[SWS_EthIf_00111] ⌈In order to access the Ethernet controller(s), the Ethernet Interface shall use one or multiple Ethernet Driver modules, which abstract the specific features and interfaces of the respective Ethernet controller(s).⌋

[SWS_EthIf_00123] ⌈In order to access the Ethernet transceiver(s), the Ethernet Interface shall use one or multiple Ethernet Transceiver Driver modules, which abstract the specific features and interfaces of the respective Ethernet transceiver(s).⌋

[SWS_EthIf_00228] ⌈In order to access the Ethernet switch(es), the Ethernet Interface shall use one or multiple Ethernet Switch Driver modules, which abstract the specific features and interfaces of the respective Ethernet switch(es).⌋

[SWS_EthIf_00112] ⌈Therefore, the Ethernet Interface executable code (however, not the configuration used during runtime) shall be completely independent of the Ethernet Communication Controller(s).⌋

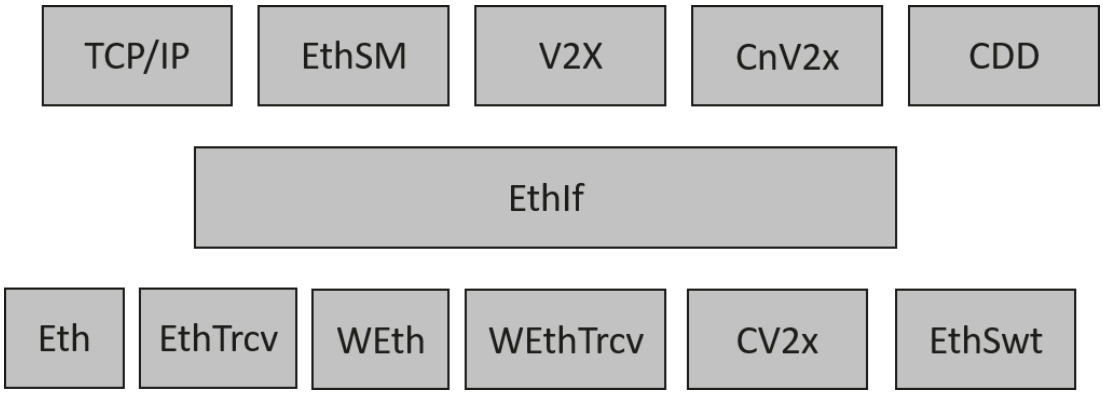

Figure 1.1: Ethernet stack module overview

Note: The Ethernet Interface is specified in a way that allows for object code delivery of the code module, following the "one-fits-all" principle, i.e. the entire configuration of the Ethernet Interface can be carried out without modifying any source code. Thus, the configuration of the Ethernet Interface can be carried out largely without detailed knowledge of the underlying hardware.

2 Acronyms and Abbreviations

The glossary below includes acronyms and abbreviations relevant to the Ethernet Interface module that are not included in the [4, AUTOSAR glossary].

Abbreviation / Acronym:	Description:
CBR	Channel Busy Ratio
CIT	Channel Idle Time
CV2x	Cellular Vehicle to X driver
Eth	Ethernet Controller Driver (AUTOSAR BSW module)
EthIf	Ethernet Interface (AUTOSAR BSW module)
EthSM	Ethernet State Manager (AUTOSAR BSW module)
EthTrcv	Ethernet Transceiver Driver (AUTOSAR BSW module)
IP	Internet Protocol
MCG	Module Configuration Generator
MII	Media Independent Interface (standardized Interface provided by Ethernet controllers to access Ethernet transceivers)
RSSI	Received Signal Strength Indicator
TCP	Transmission Control Protocol
TCP/IP Stack	Ethernet communication stack
VLAN	Virtual Local Area Network
WEth	Wireless Ethernet Driver
WEthTrcv	Wireless Ethernet Transceiver Driver
OA TC10	Open Alliance TC10 Specification [5]

Table 2.1: Acronyms and Abbreviations

3 Related documentation

3.1 Input documents & related standards and norms

[1] Layered Software Architecture
AUTOSAR_CP_EXP_LayeredSoftwareArchitecture

[2] Specification of TCP/IP Stack
AUTOSAR_CP_SWS_TcpIp

[3] Specification of Ethernet State Manager
AUTOSAR_CP_SWS_EthernetStateManager

[4] Glossary
AUTOSAR_FO_TR_Glossary

[5] OPEN Sleep/Wake-up Specification for Automotive Ethernet
http://www.opensig.org/Automotive-Ethernet-Specifications/

[6] General Specification of Basic Software Modules
AUTOSAR_CP_SWS_BSWGeneral

[7] Specification of Vehicle-2-X Geo Networking
AUTOSAR_CP_SWS_V2XGeoNetworking

[8] Specification of Chinese Vehicle-2-X Network
AUTOSAR_CP_SWS_ChineseV2XNetwork

[9] Specification of Chinese Vehicle-2-X Management
AUTOSAR_CP_SWS_ChineseV2XManagement

[10] Specification of Ethernet Driver
AUTOSAR_CP_SWS_EthernetDriver

[11] Specification of Ethernet Transceiver Driver
AUTOSAR_CP_SWS_EthernetTransceiverDriver

[12] General Requirements on Basic Software Modules
AUTOSAR_CP_RS_BSWGeneral

[13] Requirements on Ethernet Support in AUTOSAR
AUTOSAR_CP_RS_Ethernet

[14] Specification of Default Error Tracer
AUTOSAR_CP_SWS_DefaultErrorTracer

[15] Specification of Time Synchronization over Ethernet
AUTOSAR_CP_SWS_TimeSyncOverEthernet

[16] Specification of Wireless Ethernet Driver
AUTOSAR_CP_SWS_WirelessEthernetDriver

[17] Specification of IEEE1722 Transport Protocol Module

AUTOSAR_CP_SWS_IEEE1722TransportLayer

[18] Specification of Linklayer Sdu Routing Module
AUTOSAR_CP_SWS_LSduRouter

[19] IEEE 802.3-2022
https://www.ieee802.org/3/

[20] Specification of Ethernet Switch Driver
AUTOSAR_CP_SWS_EthernetSwitchDriver

[21] Specification of Wireless Ethernet Transceiver Driver
AUTOSAR_CP_SWS_WirelessEthernetTransceiverDriver

[22] Specification of Cellular Vehicle-2-X Driver
AUTOSAR_CP_SWS_CellularV2XDriver

[23] IEEE Standard for Local and Metropolitan Area Networks–Port-Based Network Access Control
https://ieeexplore.ieee.org/document/9018454

[24] IEEE Standard for Local and metropolitan area networks-Media Access Control (MAC) Security
https://ieeexplore.ieee.org/document/8585421

3.2 Related specification

AUTOSAR provides a General Specification on Basic Software modules [6, SWS BSW General], which is also valid for Ethernet Interface.

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

4 Constraints and assumptions

4.1 Limitations

The Ethernet Interface is conceptually able to access:

- one or more Ethernet Driver
- one or more Ethernet Transceiver Driver
- one or more Ethernet Switch Driver

AUTOSAR supports for MACsec in hardware a hardware topology where the Ethernet controller has direct access to the destination hardware via one type of hardware management interface (e.g. MDIO). Thus, AUTOSAR supports MACsec in hardware only where the affected hardware is directly reachable by the Ethernet controller. Or in other words, AUTOSAR does not support a hardware topology where control commands to the destination hardware needs to be transferred across multiple management interfaces of a different type (e.g. MDIO to SPI and vs.). For example, if an Ethernet controller connects an Ethernet switch via SPI and the Ethernet switch connects external PHYs via MDIO, then the following constrains need to be respected:

- MACsec support performed by the Ethernet controller is supported
- MACsec support performed by the Ethernet switch port is supported
- MACsec support performed by the external PHY connected to the Ethernet switch is not supported

It is not possible to transmit data which exceeds the available buffer size of the used Ethernet controller. Longer data has to be transmitted using the Internet Protocol (IP) or Transmission Control Protocol (TCP).

4.2 Applicability to car domains

The Ethernet BSW stack is intended to be used wherever high data rates are required but no hard real-time is required. Of course, it can also be used for less-demanding use cases, i.e. for low data rates.

5 Dependencies to other modules

This chapter lists the modules interacting with the Ethernet Interface module.

Modules that use Ethernet Interface module:

- Ethernet Communication Stack (TCP/IP Stack [2])
- Ethernet State Manager (EthSM) [3]
- V2xGn [7]
- CnV2xNet [8]
- CnV2xM [9]

Dependencies to other Modules:

- The Ethernet Interface module doesn't take care of configuring Ethernet Driver [10] but requires its preceding initialization and configuration.
- The Ethernet Interface module doesn't take care of configuring Ethernet Transceiver Driver [11] but requires its preceding initialization and configuration.

6 Requirements Tracing

The following tables reference the requirements specified in [12, SRS BSWGeneral] and [13, SRS Ethernet] and links to the fulfillment of these. Please note that if column “Satisfied by” is empty for a specific requirement this means that this requirement is not fulfilled by this document.

Requirement	Description	Satisfied by
[FO_RS_Fw_00011]	Hardware-Accelerated Filtering Support	[SWS_EthIf_91023] [SWS_EthIf_91024] [SWS_EthIf_91025] [SWS_EthIf_91027]
[FO_RS_MACsec_ 00001]	MACsec Protocol support	[SWS_EthIf_00560]
[FO_RS_MACsec_ 00002]	MACsec Key Agreement Protocol support	[SWS_EthIf_00682] [SWS_EthIf_00684] [SWS_EthIf_00685] [SWS_EthIf_00686] [SWS_EthIf_00687] [SWS_EthIf_00688] [SWS_EthIf_00689] [SWS_EthIf_00690] [SWS_EthIf_00691] [SWS_EthIf_00692] [SWS_EthIf_00693] [SWS_EthIf_00694] [SWS_EthIf_00695] [SWS_EthIf_00696]
[FO_RS_MACsec_ 00004]	Configure which Ethernet ports use MACsec	[SWS_EthIf_00561] [SWS_EthIf_00562] [SWS_EthIf_91253] [SWS_EthIf_91254] [SWS_EthIf_91255] [SWS_EthIf_91256]
[FO_RS_MACsec_ 00007]	Configuration of unprotected traffic (for Software-based MACsec)	[SWS_EthIf_00563]
[FO_RS_MACsec_ 00009]	MACsec Security Events	[SWS_EthIf_00564]
[FO_RS_MACsec_ 00010]	Support of integrity and confidentiality	[SWS_EthIf_00565]
[FO_RS_MACsec_ 00011]	MAC Security TAG	[SWS_EthIf_00566] [SWS_EthIf_00568] [SWS_EthIf_00569] [SWS_EthIf_00570] [SWS_EthIf_00571]
[FO_RS_MACsec_ 00012]	MACsec EtherType	[SWS_EthIf_00567]
[FO_RS_MACsec_ 00017]	Support of Extended Packet Number (XPN)	[SWS_EthIf_00572]
[FO_RS_MACsec_ 00018]	Secure Channel Identifier (SCI)	[SWS_EthIf_00573]
[FO_RS_MACsec_ 00019]	Secure Data	[SWS_EthIf_00574]
[FO_RS_MACsec_ 00020]	Integrity Check Value (ICV)	[SWS_EthIf_00575]
[FO_RS_MACsec_ 00021]	Protect function in software solution	[SWS_EthIf_00576]
[FO_RS_MACsec_ 00022]	Validation function in software solution	[SWS_EthIf_00577]
[FO_RS_MACsec_ 00023]	Support of MKA Packets	[SWS_EthIf_00583]
[FO_RS_MACsec_ 00032]	List of minimal supported cipher suites	[SWS_EthIf_00578]
[FO_RS_MACsec_ 00033]	Validation function for ICVs	[SWS_EthIf_00579]
[FO_RS_MACsec_ 00034]	Generation function for ICVs	[SWS_EthIf_00580]

△

Requirement	Description	Satisfied by
[RS_Ids_00810]	Basic SW security events	[SWS_EthIf_00502] [SWS_EthIf_00503] [SWS_EthIf_00698] [SWS_EthIf_00699] [SWS_EthIf_00700] [SWS_EthIf_00701] [SWS_EthIf_00702] [SWS_EthIf_00703] [SWS_EthIf_00704] [SWS_EthIf_00705]
[SRS_BSW_00170]	The AUTOSAR SW Components shall provide information about their dependency from faults, signal qualities, driver demands	[SWS_EthIf_00999]
[SRS_BSW_00171]	Optional functionality of a Basic-SW component that is not required in the ECU shall be configurable at pre-compile-time	[SWS_EthIf_00605] [SWS_EthIf_00610] [SWS_EthIf_00623] [SWS_EthIf_00630] [SWS_EthIf_00635]
[SRS_BSW_00323]	All AUTOSAR Basic Software Modules shall check passed API parameters for validity	[SWS_EthIf_00652] [SWS_EthIf_00653] [SWS_EthIf_00654] [SWS_EthIf_00655] [SWS_EthIf_00656] [SWS_EthIf_00657] [SWS_EthIf_00658] [SWS_EthIf_00679]
[SRS_BSW_00337]	Classification of development errors	[SWS_EthIf_00652] [SWS_EthIf_00653] [SWS_EthIf_00654] [SWS_EthIf_00655] [SWS_EthIf_00656] [SWS_EthIf_00657] [SWS_EthIf_00658] [SWS_EthIf_00679]
[SRS_BSW_00350]	All AUTOSAR Basic Software Modules shall allow the enabling/ disabling of detection and reporting of development errors.	[SWS_EthIf_00600] [SWS_EthIf_00638] [SWS_EthIf_00639] [SWS_EthIf_00640] [SWS_EthIf_00641] [SWS_EthIf_00644] [SWS_EthIf_00645] [SWS_EthIf_00646] [SWS_EthIf_00647] [SWS_EthIf_00651] [SWS_EthIf_00663] [SWS_EthIf_00664] [SWS_EthIf_00665] [SWS_EthIf_00666] [SWS_EthIf_00667] [SWS_EthIf_00670] [SWS_EthIf_00671] [SWS_EthIf_00672] [SWS_EthIf_00673] [SWS_EthIf_00674] [SWS_EthIf_00675] [SWS_EthIf_00676] [SWS_EthIf_00677] [SWS_EthIf_00678]
[SRS_BSW_00385]	List possible error notifications	[SWS_EthIf_91136]
[SRS_BSW_00386]	The BSW shall specify the configuration and conditions for detecting an error	[SWS_EthIf_00600] [SWS_EthIf_00638] [SWS_EthIf_00639] [SWS_EthIf_00640] [SWS_EthIf_00641] [SWS_EthIf_00644] [SWS_EthIf_00645] [SWS_EthIf_00646] [SWS_EthIf_00647] [SWS_EthIf_00651] [SWS_EthIf_00663] [SWS_EthIf_00664] [SWS_EthIf_00665] [SWS_EthIf_00666] [SWS_EthIf_00667] [SWS_EthIf_00670] [SWS_EthIf_00671] [SWS_EthIf_00672] [SWS_EthIf_00673] [SWS_EthIf_00674] [SWS_EthIf_00675] [SWS_EthIf_00676] [SWS_EthIf_00677] [SWS_EthIf_00678]
[SRS_BSW_00450]	A Main function of a un-initialized module shall return immediately	[SWS_EthIf_00651]
[SRS_BSW_00459]	It shall be possible to concurrently execute a service offered by a BSW module in different partitions	[SWS_EthIf_00606] [SWS_EthIf_00611] [SWS_EthIf_00625] [SWS_EthIf_00632]
[SRS_BSW_00461]	Modules called by generic modules shall satisfy all interfaces requested by the generic module	[SWS_EthIf_00681]

▽

△

Requirement	Description	Satisfied by
[SRS_Eth_00033]	The Ethernet Interface shall provide indication for link state change.	[SWS_EthIf_00680] [SWS_EthIf_00682] [SWS_EthIf_00683] [SWS_EthIf_00684] [SWS_EthIf_00685] [SWS_EthIf_00686] [SWS_EthIf_00687] [SWS_EthIf_00689] [SWS_EthIf_00697]
[SRS_Eth_00106]	The Ethernet Transceiver Driver shall switch on/off wake up functionality at pre compile time.	[SWS_EthIf_00245] [SWS_EthIf_00500]
[SRS_Eth_00107]	The Ethernet Transceiver Driver shall support access to the wake up reason.	[SWS_EthIf_00486] [SWS_EthIf_00490] [SWS_EthIf_91004]
[SRS_Eth_00117]	The Ethernet Transceiver Driver shall provide access to standardized hardware features	[SWS_EthIf_00474] [SWS_EthIf_91014] [SWS_EthIf_91016] [SWS_EthIf_91018] [SWS_EthIf_91020] [SWS_EthIf_91021] [SWS_EthIf_91061]
[SRS_Eth_00120]	Hardware access via MII and/or SPI	[SWS_EthIf_91249] [SWS_EthIf_91250]
[SRS_Eth_00125]	The Ethernet Switch Driver shall support switch frame management	[SWS_EthIf_00201] [SWS_EthIf_00202] [SWS_EthIf_00203] [SWS_EthIf_00204] [SWS_EthIf_91003] [SWS_EthIf_91007] [SWS_EthIf_91243] [SWS_EthIf_91244] [SWS_EthIf_91245] [SWS_EthIf_91246] [SWS_EthIf_91247] [SWS_EthIf_91248]
[SRS_Eth_00156]	The Ethernet Interface shall provide indication for a received sleep request.	[SWS_EthIf_00497] [SWS_EthIf_00499] [SWS_EthIf_91006]
[SRS_Eth_00157]	The Ethernet Interface shall trigger requested modes for Ethernet hardware with wake-up capability even if the requested mode has already been reached.	[SWS_EthIf_00264] [SWS_EthIf_00266] [SWS_EthIf_00478] [SWS_EthIf_00479] [SWS_EthIf_00480] [SWS_EthIf_00481] [SWS_EthIf_00482] [SWS_EthIf_00483] [SWS_EthIf_00504] [SWS_EthIf_00649] [SWS_EthIf_00650]
[SRS_Eth_00169]	Ethernet Interface upper layer PDU based communication	[SWS_EthIf_00085] [SWS_EthIf_91138]
[SRS_Eth_00170]	Ethernet Interface scheduling a subset of ingress queues	[SWS_EthIf_00648] [SWS_EthIf_91139]
[SRS_Eth_00175]	The Ethernet Interface shall support access to PTP Physical Clocks	[SWS_EthIf_00585] [SWS_EthIf_00586] [SWS_EthIf_00624] [SWS_EthIf_00631] [SWS_EthIf_00636] [SWS_EthIf_91062] [SWS_EthIf_91063] [SWS_EthIf_91064] [SWS_EthIf_91066]
[SRS_Eth_00176]	The Ethernet Interface shall support control of pulse per second signal generation	[SWS_EthIf_91065]
[SRS_Eth_00182]	The Ethernet Interface shall support hardware independent APIs to access hardware functionality and configuration via the Ethernet stack drivers	[SWS_EthIf_00659] [SWS_EthIf_00660]

Table 6.1: Requirements Tracing

7 Functional specification

7.1 Ethernet BSW stack

As part of the AUTOSAR Layered Software Architecture [1], the Ethernet BSW modules also form a layered software stack. Figure 7.1 depicts the basic structure of this Ethernet BSW stack. The Ethernet Interface module accesses several Ethernet controllers using the Ethernet Driver layer, which can be made up of several Ethernet Drivers modules.

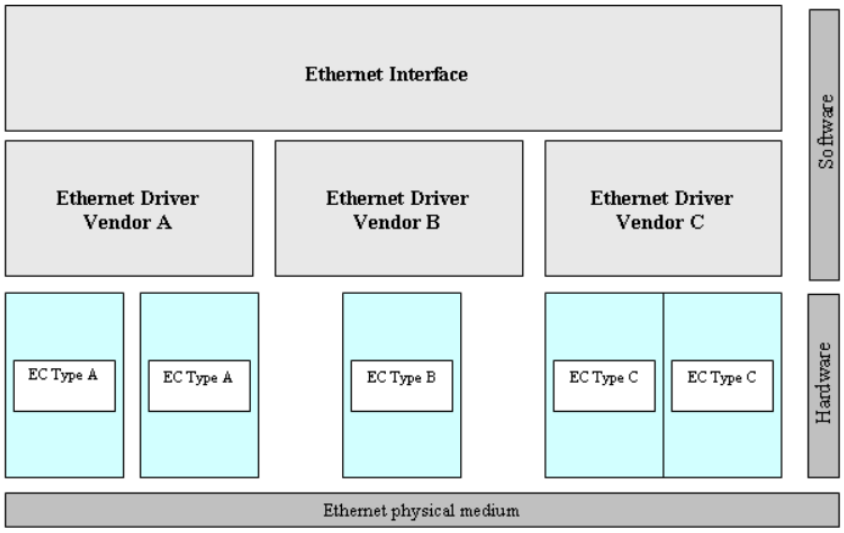

Figure 7.1: Basic Structure of the Enthernet BSW Stack

7.1.1 Indexing scheme for Ethernet controller

In case CAN XL is used as physical medium, the configuration will contain an EthIfEthCanXLCtrlRef instead of an EthIfEthCtrlRef and an EthIfCanXLTrcvRef instead of an EthIfEthTrcvRef. In this case, APIs denoted as <EthDrv>_Xxx will be called as CanXL_Xxx, otherwise as Eth_Xxx, and likewise APIs denoted as <EthTrcv>_Yyy will be called as CanXLTrcv_Yyy, otherwise EthTrcv_Yyy.

Users of the Ethernet Interface identify Ethernet controller resources using an indexing scheme as depicted in Figure 7.2.

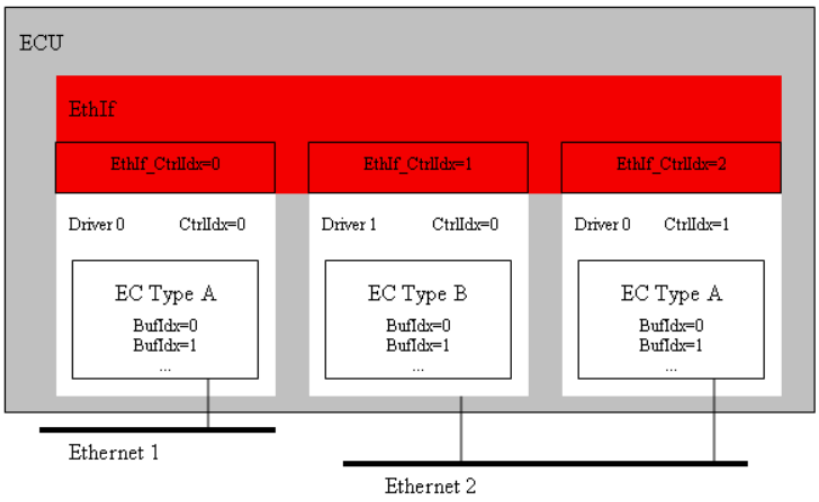

Figure 7.2: Ethernet Interface controller indexing scheme

[SWS_EthIf_00003] ⌈The Ethernet Interface is using an index (EthIfCtrlIdx) to abstract the access to VLANs from the underlying communication system compromised of Ethernet Controller and Ethernet Transceiver.

Therefore the Ethernet Interface shall implement a mapping from Ethernet Interface controllers (EthIfCtrlIdx) to respective hardware ressource controllers (EthCtrlId + EthTrcvId).⌋

7.1.2 Indexing scheme for Ethernet switches

Since the EthIf is not concerned with the individual EthSwtPorts which belong to the individual EthSwtes there is no indexing scheme for EthSwtPorts required in the EthIf. Any BSW module which interacts with EthSwtPorts can directly refer to the ECU configuration of the EthSwtPort for the indexing.

[SWS_EthIf_00224] ⌈The EthIf shall dispatch all accesses by the EthIfSwitchIdx index to the respective EthSwt driver module with the EthSwtIdx value⌋

7.1.3 Initialization

The `EthIf` module is initialized via `EthIf_Init`, and de-initialized via `EthIf_DeInit`. Except for `EthIf_GetVersionInfo`, `EthIf_Init` and any `EthIf` scheduled function (e.g. `EthIf_MainFunctionTx`), the API functions of the `EthIf` module may only be called after the module has been properly initialized.

[SWS_EthIf_00651] DET error reporting of `ETHIF_E_UNINIT`

Upstream requirements: SRS_BSW_00350, SRS_BSW_00386, SRS_BSW_00450

⌈If development error reporting is enabled via `EthIfDevErrorDetect`, the `EthIf` module shall call `Det_ReportError` with the error code `ETHIF_E_UNINIT` when any API other than `EthIf_Init`, `EthIf_GetVersionInfo` or any `EthIf` scheduled function (e.g. `EthIf_MainFunctionTx`) is called in uninitialized state.⌋

7.1.4 Ethernet Interface main function

[SWS_EthIf_00004] ⌈The Ethernet Interface shall implement main functions to be used for frame transmission confirmation and frame reception in polling mode with a calling period configurable at system configuration time.⌋

7.1.5 Requirements

This chapter lists requirements that shall be fulfilled by Ethernet Interface module implementations.

The Ethernet Interface module environment comprises all modules which are calling interfaces of the Ethernet Interface module.

[SWS_EthIf_00005] ⌈The Ethernet Interface module shall support pre-compile time, link time and post-build time configuration.⌋

[SWS_EthIf_00006] ⌈The header file EthIf.h shall include a software and specification version number.⌋

[SWS_EthIf_00007] ⌈The Ethernet Interface module shall perform a consistency check between code files and header files based on pre-process-checking the version numbers of related code files and header files.⌋

DET API functions are specified in [14, Specification of Default Error Tracer].

[SWS_EthIf_00010] ⌈The Ethernet Interface module shall implement the API functions specified by the Ethernet Interface SWS as real C-code functions and shall not implement the API as macros for object code deliveries.⌋

[SWS_EthIf_00011] ⌈None of the Ethernet Interface module header files shall define global variables.⌋

7.1.6 Configuration description

[SWS_EthIf_00012] ⌈The Ethernet Interface module shall provide an XML file that contains the data, which is required for the SW identification (it shall contain the vendor identification, module ID and software version information), configuration and integration process. This file should describe vendor specific configuration parameters as well as it should contain recommended configuration parameter values.⌋

[SWS_EthIf_00117] ⌈The MCG shall read the ECU configuration description of the Ethernet Driver and the Ethernet Interface module(s). While cluster related configuration parameters are contained in the Ethernet Interface module configuration description, Ethernet Driver related configuration data is contained in the Ethernet Driver module configuration description. The Ethernet Interface module specific configuration tool shall read both ECU module descriptions to derive the configuration data for all Ethernet Drivers mapped to the Ethernet Interface module.⌋

[SWS_EthIf_00118] ⌈The MCG shall ensure the consistency of the generated configuration data.⌋

[SWS_EthIf_00013] ⌈The configuration of the Ethernet Interface module shall be configured at ECU configuration time. None of the communication parameters shall be configured at runtime.⌋

[SWS_EthIf_00014] ⌈The start address of post-build time configuration data shall be passed during module initialization.⌋

An assignment of those configuration classes to configuration parameters can be found in chapter 10.

A detailed description of all Ethernet Interface related configuration parameters can be found in chapter 10 of this document. Additionally, the configuration description of the Ethernet Driver (see chapter 10 of [10, Specification of Ethernet Driver]) shall be evaluated for Ethernet Interface module configuration.

7.1.7 VLAN support

[SWS_EthIf_00128] ⌈The Ethernet Interface shall support Virtual Local Area Networks (VLAN).⌋

[SWS_EthIf_00129] ⌈The Ethernet Interface shall encapsulate Virtual Local Area Networks (VLAN) into virtual controllers (Ethernet Interface controller) representing a dedicated VLAN.

All BSW modules above the Ethernet Interface shall interact based on those virtual controllers.

The Ethernet Driver and Transceiver deal only with real controllers and are not aware of the existence of virtual controllers.

Caveat: the virtual controller represents the untagged VLAN if no VLAN ID is set.⌋

[SWS_EthIf_00130] ⌈The Ethernet Interface shall use the buffers provided by the Ethernet Driver for VLAN support. If Can XL is used the Ethernet Interface shall use the buffers provided by the Can XL Driver.⌋

7.1.8 Wake up support

The Ethernet Interface supports wake up depending on the parameter EthIfWakeUpSupport.

Note: Enabling wake-up support in EthIf makes only sense if the underlying EthTrcv supports also wake up.

7.1.9 Ethernet Switch Management support

[SWS_EthIf_00201] Forward a call of EthIf_GetRxMgmtObject via EthSwt_GetRxMgmtObject to EthSwt

Upstream requirements: SRS_Eth_00125

⌈If `EthIf_GetRxMgmtObject` is called and the given PduId refer to an `EthIfFrameRxPdu` which is in state `PDU_IN_USE`, then EthIf shall forward the call to the Ethernet Switch via EthSwt_GetRxMgmtObject by considering the following points. Otherwise return with `E_NOT_OK`:

- set the CtrlIdx to the index of the `EthIfController`, where this `EthIfFrameRxPdu` is associated with
- set the DataPtr to `PduInfo.SduDataPtr`, which belongs to the given PduId in the context of this call
- set the MgmtObjectPtr to the given MgmtObjectPtr

⌋

[SWS_EthIf_00202] Forward a call of EthIf_GetTxMgmtObject via EthSwt_GetTxMgmtObject to EthSwt

Upstream requirements: SRS_Eth_00125

⌈If `EthIf_GetTxMgmtObject` is called and the given PduId refer to an `EthIfFrameTxPdu` which is in state `PDU_IN_USE`, then EthIf shall forward the call to the Ethernet Switch via EthSwt_GetTxMgmtObject by considering the following points. Otherwise return with `E_NOT_OK`:

- set the CtrlIdx to the index of the EthIfController, where this EthIfFrameRxPdu is associated with
- set the MgmtObjectPtr to the given MgmtObjectPtr

⌋

[SWS_EthIf_00203] Forward a call of EthIf_SetSwitchMgmtInfo via EthSwt_SetMgmtInfo to EthSwt

Upstream requirements: SRS_Eth_00125

⌈If EthIf_SetSwitchMgmtInfo is called and the given PduId refer to an EthIfFrameTxPdu which is in state PDU_IN_USE, then EthIf shall forward the call to the Ethernet Switch via EthSwt_SetMgmtInfo by considering the following points. Otherwise return with E_NOT_OK:

- set the CtrlIdx to the index of the EthIfController, where this EthIfFrameRxPdu is associated with
- set the BufIdx to the value that is associated with PduId (see [SWS_EthIf_00672])
- set the MgmtObjectPtr to the given MgmtObjectPtr

⌋

[SWS_EthIf_00204] Forward a call of EthIf_SwitchEnableTimeStamping via EthSwt_PortEnableTimeStamp to EthSwt

Upstream requirements: SRS_Eth_00125

⌈If EthIf_SwitchEnableTimeStamping is called and the given PduId refer to an EthIfFrameTxPdu which is in state PDU_IN_USE, then EthIf shall forward the call to the Ethernet Switch via EthSwt_PortEnableTimeStamp by considering the following points. Otherwise return with E_NOT_OK:

- set the CtrlIdx to the index of the EthIfController, where this EthIfFrameRxPdu is associated with
- set the BufIdx to the value that is associated with PduId (see [SWS_EthIf_00672])
- set the MgmtObjectPtr to the given MgmtObjectPtr

⌋

[SWS_EthIf_00387] ⌈If EthIf_SwitchEnableTimeStamping is called, the EthIf shall call EthSwt_PortEnableTimeStamp for every port in the group.⌋

Note: Call of EthIf_SetSwitchMgmtInfo or EthIf_SwitchEnableTimeStamping is working properly, where a transmit buffer is allocated via call of Eth_ProvideTxBuffer up front to the transmission request. This is granted for the following transmission requests:

- Transmission request with direct data provision and deferred forwarding (see [SWS_EthIf_00671])

- Transmission request with indirect data provision (see [SWS_EthIf_00676])

Ethernet switch management enables the possibility to control an Ethernet frame regarding an Ethernet switch port specific ingress and egress handling as well as providing a Ethernet switch port specific timestamp. This functionality is essential for other BSW modules, in particular for EthTSyn, which requires Port specific information associated to a time synchronization [15] or path-delay measurement frame.

For an introduction of the basic HW architecture and interaction, please refer to [10, Specification of Ethernet Driver].

For more details regarding functional sequences, please refer to [16, Specification of Wireless Ethernet Driver].

Note: Ethernet switch management API's supporting the <Upper Layer> to gather / modify Ethernet switch port specific communication attributes.

7.1.10 Handling of maintained Ethernet hardware

The Ethernet Interface handle the maintained Ethernet hardware due to its configuration:

- EthIfPhysController (representing physical Ethernet controller)
- EthIfController (representing virtual Ethernet controller to support VLANs)
- EthIfTransceiver (representing PHYs)
- EthIfSwitch (representation of an Ethernet switch)
- EthIfSwitchPortGroups (representing groups of EthSwtPorts)

At least one EthIfPhysController should be present in the configuration to interact with the Ethernet driver. EthIfController represent the connection between the physical Ethernet controller and used Ethernet hardware to communicate on and Ethernet network. This could be either an EthIfTransceiver or an EthIfSwitch or an EthIfSwitchPortGroup. If an upper layer wants to control the communication on a particular Ethernet network, it calls the corresponding EthIfController via `EthIf_SetControllerMode`. The Ethernet Interface handle a communication request, such that it takes care to forward the request to the corresponding Ethernet hardware:

- EthIfTransceiver
- EthIfSwitch
- EthIfSwitchPortGroup with reference of type "control"

For EthIfController with reference of type "link-information" to an EthIfSwitchPortGroup, the Ethernet Interface supervise the link state of all EthSwtPorts within a EthIfSwitchPortGroup and signal the accumulated link state to the corresponding upper layer (EthSM [3]). Those EthIfSwitchPortGroups are controlled via a call of `EthIf_Switch-`

PortGroupRequestMode. This is used if EthIfSwitchPortGroups are controlled according to partial network requests. Partial network requests are forwarded to BswM and a particular rule in the BswM lead to an action to control the corresponding EthIfSwitchPortGroup. Thus the upper layer of the Ethernet Interface to control the communication is EthSM and the BswM, if EthIfSwitchPortGroup switching is used. Independent if an EthIfController or an EthIfSwitchPortGroup are addressed for a communication request, the upper layer request the Ethernet Connection to be ACTIVE (ETH_MODE_ACTIVE or ETH_MODE_ACTIVE_WITH_WAKEUP_REQUEST) or DOWN (ETH_MODE_DOWN). The Ethernet Inteface requests the corresponding lower layer to switch on the corresponding Ethernet hardware for an ACTIVE-request or switch off the corresponding Ethernet Hardware for a DOWN-request.

7.1.10.1 EthIfSwitchPortGroup handling

The Ethernet Interface supports the grouping of Ethernet switch ports (EthIfSwitchPortGroup). The request (either ACTIVE or DOWN) will be handled and rated by the Ethernet Interface. The Ethernet Interface has to decide either to put the EthifSwitchPotGroup to DOWN or ACTIVE state. ACTIVE-request for EthIfSwitchPortGroup will always overrule DOWN-request for EthIfSwitchPortGroups. If a DOWN-request for an EthIfSwitchPortGroup is ready for execution, the EthIf will check the EthSwtPorts which are referenced by the EthIfSwitchPortGroup and decide if the EthSwtPort can be set to DOWN state. If this is valid, the EthSwtPort is set to DOWN state after the configured switch off delay timer has expired.

Note: Further requirements for switching of EthIfSwitchPortGroups are available in chapter 7.1.10.2 and 8.4.5.5.

7.1.10.2 Switching of EthIfController and the corresponding Ethernet hardware

Switching of an EthIfController is triggered via a call of EthIf_SetControllerMode. Switching of an EthIfController implicitly include the switching of the corresponding Ethernet hardware (PHY, Ethernet switch, Ethernet switch port). The Ethernet Interface interact with the lower layer via asynchronous callback notification (e.g. EthIf_TrcvModeIndication). The chapter describe the interaction of the APIs used to switch the EthIfController and the corresponding Ethernet hardware.

Note:

1. A call of the EthIf_SetControllerMode causes an asynchronous indication by calling EthIf_CtrlModeIndication, if the mode of the referenced EthIfPhysController has changed.
2. The requirements assume that Ethernet Controller (EthIfPhysControllerIdx) and the referenced Ethernet hardware (e.g. PHY, Ethernet Switch) are controlled independent from each other. For example, if ETH_MODE_ACTIVE or ETH_MODE_

ACTIVE_WITH_WAKEUP_REQUEST has been requested and Ethernet Controller Driver of the affected Ethernet Controller (EthIfPhysControllerIdx) has NOT indicated ETH_MODE_ACTIVE yet, then those requests can be forwarded directly to the corresponding lower layers of the referenced Ethernet hardware. An implementation has to consider the following points:

- ETH_MODE_ACTIVE and ETH_MODE_DOWN are activating and de-activiating the communication capability of an Ethernet Controller, but not the control capability of connected Ethernet hardware (e.g. MDIO).
- The implementation has to ensure, that the control capabilities via an Ethernet controller are always available, if needed by the driver modules (e.g. Ethernet switch driver)

3. EthIf has to ensure that a request with ETH_MODE_ACTIVE_WITH_WAKEUP_REQUEST is not overwritten by another call of EthIf_SetControllerMode with ETH_MODE_ACTIVE, if the request is deferred due to the EthIfPhysController has not already indicated ETH_MODE_ACTIVE.

[SWS_EthIf_00035] ⌈The function EthIf_SetControllerMode shall forward the call to function <EthDrv>_SetControllerMode of the corresponding Ethernet Controller Driver (EthIfPhysControllerIdx) with ETH_MODE_ACTIVE, if mode ETH_MODE_ACTIVE or ETH_MODE_ACTIVE_WITH_WAKEUP_REQUEST has been requested and the corresponding Ethernet Controller Driver (EthIfPhysControllerIdx) has NOT already indicated ETH_MODE_ACTIVE.⌋

[SWS_EthIf_00266]

Upstream requirements: SRS_Eth_00157

⌈If EthIf_SetControllerMode has been called for an EthIfController with ETH_MODE_ACTIVE and this EthIfController has a reference to an EthIfTransceiver, then EthIf shall forward the call to the following functions in the given order, if the current mode of the EthIfTransceiver is ETH_MODE_DOWN:

1. <EthTrcv>_SetTransceiverMode with ETH_MODE_ACTIVE
2. <EthTrcv>_TransceiverLinkStateRequest with ETHTRCV_LINK_STATE_ACTIVE

⌋

[SWS_EthIf_00478]

Upstream requirements: SRS_Eth_00157

⌈If EthIf_SetControllerMode has been called for an EthIfController with ETH_MODE_ACTIVE and this EthIfController has a reference to an EthIfSwitch, then EthIf shall forward the call to the following functions in the given order for all EthSwtPorts of the referenced switch if mode ETH_MODE_ACTIVE has been requested and the current EthSwtPort mode is ETH_MODE_DOWN:

1. EthSwt_SetSwitchPortMode with ETH_MODE_ACTIVE

2. EthSwt_PortLinkStateRequest with ETHTRCV_LINK_STATE_ACTIVE

⌋

[SWS_EthIf_00264]

Upstream requirements: SRS_Eth_00157

⌈If EthIf_SetControllerMode has been called for an EthIfController with ETH_MODE_ACTIVE and this EthIfController has a reference to an EthIfSwitchPortGroup of type "control", then EthIf shall forward the call to the following functions in the given order for all EthSwtPorts of the respective EthIfSwitchPortGroup if the mode ETH_MODE_ACTIVE has been requested for the first EthIfSwitchPortGroup referencing the EthSwtPort and the current EthSwtPort mode is ETH_MODE_DOWN:

1. EthSwt_SetSwitchPortMode with ETH_MODE_ACTIVE
2. EthSwt_PortLinkStateRequest with ETHTRCV_LINK_STATE_ACTIVE

⌋

Note: EthIfController that reference EthIfSwitfhPortGroups and the reference is of type "link-information" (see [ECUC_EthIf_00048]), then those EthIfSwitchPortGroups could be switched according to PNC states via a dedicatd rules in the BswM. The BswM rule can be configured via the BswMEthIfSwitchPortGroupRequestMode action. The BswM call the API EthIf_SwitchPortGroupRequestMode to switch the corresponding EthIfSwitchPortGroup.

[SWS_EthIf_00272] ⌈If EthIf_SwitchPortGroupRequestMode has been called with ETH_MODE_ACTIVE, EthIf shall forward the call to the following functions in the given order for all EthSwtPorts of the respective EthIfSwitchPortGroup:

1. Call EthSwt_SetSwitchPortMode with ETH_MODE_ACTIVE, if the current mode is ETH_MODE_DOWN.
2. Call EthSwt_PortLinkStateRequest with ETHTRCV_LINK_STATE_ACTIVE, if the current link state is ETHTRCV_LINK_STATE_DOWN

⌋

[SWS_EthIf_00479]

Upstream requirements: SRS_Eth_00157

⌈Everytime EthIf_SetControllerMode has been called for an EthIfController with ETH_MODE_ACTIVE_WITH_WAKEUP_REQUEST and this EthIfController has a reference to an EthIfTransceiver, then EthIf shall forward the call to the following functions in the given order, independent of the current mode:

1. <EthTrcv>_SetTransceiverMode with ETH_MODE_ACTIVE_WITH_WAKEUP_REQUEST
2. <EthTrcv>_TransceiverLinkStateRequest with ETHTRCV_LINK_STATE_ACTIVE, only if the current state is ETHTRCV_LINK_STATE_DOWN

⌋

[SWS_EthIf_00480]

Upstream requirements: SRS_Eth_00157

⌈Everytime `EthIf_SetControllerMode` has been called for an EthIfController with `ETH_MODE_ACTIVE_WITH_WAKEUP_REQUEST` and this EthIfController has a reference to an EthIfSwitch, then EthIf shall forward the call to the following functions in the given order for all EthSwtPorts of the respective EthIfSwitchPortGroup, independ of the current mode:

1. `EthSwt_SetSwitchPortMode` with `ETH_MODE_ACTIVE_WITH_WAKEUP_REQUEST`
2. `EthSwt_PortLinkStateRequest` with `ETHTRCV_LINK_STATE_ACTIVE`, if the current mode is `ETHTRCV_LINK_STATE_DOWN`

⌋

[SWS_EthIf_00481]

Upstream requirements: SRS_Eth_00157

⌈Everytime `EthIf_SetControllerMode` has been called for an EthIfController with `ETH_MODE_ACTIVE_WITH_WAKEUP_REQUEST` and this EthIfController has a reference to an EthIfSwitchPortGroup of type "control", then EthIf shall forward the call to the following functions in the given order for all EthSwtPorts of the respective EthIfSwitchPortGroup, independent of the current mode:

1. `EthSwt_SetSwitchPortMode` with `ETH_MODE_ACTIVE_WITH_WAKEUP_REQUEST`
2. `EthSwt_PortLinkStateRequest` with `ETHTRCV_LINK_STATE_ACTIVE`, if the current mode is `ETHTRCV_LINK_STATE_DOWN`

⌋

[SWS_EthIf_00482]

Upstream requirements: SRS_Eth_00157

⌈Everytime `EthIf_SwitchPortGroupRequestMode` has been called with `ETH_MODE_ACTIVE_WITH_WAKEUP_REQUEST`, EthIf shall forward the call for all EthSwtPorts of the respective EthIfSwitchPortGroup to the following functions in the given order independent of the current EthSwtPort mode:

1. `EthSwt_SetSwitchPortMode` with `ETH_MODE_ACTIVE_WITH_WAKEUP_REQUEST`
2. `EthSwt_PortLinkStateRequest` with `ETHTRCV_LINK_STATE_ACTIVE`, only if current link state is `ETHTRCV_LINK_STATE_DOWN`

⌋

Rational for [SWS_EthIf_00479], [SWS_EthIf_00480], [SWS_EthIf_00481] and [SWS_EthIf_00482]: A wake-up request has always to be forwarded to the lower layer independent of the current mode to ensure that a wake-up is triggered on the network. This could be used for e.g. communication channels where the Ethernet hardware is compliant to OA TC10 (see [5, OPEN Sleep/Wake-up Specification for Automotive Ethernet])

[SWS_EthIf_00483]

Upstream requirements: SRS_Eth_00157

⌈If `EthIf_SwitchPortGroupRequestMode` is called with `ETH_MODE_ACTIVE` or `ETH_MODE_ACTIVE_WITH_WAKEUP_REQUEST`, then a running timer to delay the switch off all ports of the respective EthIfSwitchPortGroup (`PortGroupIdx`) shall be canceled.⌋

[SWS_EthIf_00263] ⌈EthIf shall call the function `<EthDrv>_SetControllerMode` of the corresponding Ethernet Controller Driver (EthIfPhysControllerIdx) with `ETH_MODE_DOWN`, if `EthIf_SetControllerMode` has been called with mode `ETH_MODE_DOWN` for all Ethernet Interface Controller referencing the Ethernet Controller.⌋

Note:

- In case of VLAN support, EthIf has to store internally the state of each EthIfController in order to filter out the requests from upper layers and disable the callouts to upper layers when the EthIfController is disabled.

[SWS_EthIf_00484] ⌈If `EthIf_SetControllerMode` is called for an EthIfController with `ETH_MODE_DOWN` and this EthIfController has a reference to an EthIfTransceiver, then EthIf shall forward the call to the following functions in the given order, if the current mode of the EthIfTransceiver is `ETH_MODE_ACTIVE`:

1. `<EthTrcv>_SetTransceiverMode` with `ETH_MODE_DOWN`
2. `<EthTrcv>_TransceiverLinkStateRequest` with `ETHTRCV_LINK_STATE_DOWN`

⌋

[SWS_EthIf_00485] ⌈If `EthIf_SetControllerMode` is called for an EthIfController with `ETH_MODE_DOWN` and this EthIfController has a reference to an EthIfSwitch, then EthIf shall forward the call to the following functions in the given order for all EthSwtPorts, where the current mode of the EthSwtPort is `ETH_MODE_ACTIVE`:

1. `EthSwt_PortLinkStateRequest` with `ETHTRCV_LINK_STATE_DOWN`
2. `EthSwt_SetSwitchPortMode` with `ETH_MODE_DOWN`

⌋

[SWS_EthIf_00265] ⌈If `EthIf_SetControllerMode` is called for an EthIfController with `ETH_MODE_DOWN` and this EthIfController has a reference to an EthIfSwitchPort-

Group of type "control", then EthIf shall forward the call to the following functions in the given order for all EthSwtPorts of the respective EthIf_SwitchPortGroup, but only for those EthSwtPorts where all referencing EthIfSwitchPortGroups has been requested with ETH_MODE_DOWN and the current mode of the EthSwtPort is ETH_MODE_ACTIVE:

1. EthSwt_PortLinkStateRequest with ETHTRCV_LINK_STATE_DOWN
2. EthSwt_SetSwitchPortMode with ETH_MODE_DOWN

⌋

Rationale: In case the respective EthIfController has no reference to an EthIf_SwitchPortGroup or the reference is of type "link information" the requested modes are not forwarded. This EthIf_SwitchPortGroups will be requested by an upper layer (e.g. BswM) with API EthIf_SwitchPortGroupRequestMode.

[SWS_EthIf_00661] Setting of TrcvLinkState in configured state change function when the referenced controller is not referencing a transceiver, nor a switch or switch port group ⌈If EthIf_CtrlModeIndication is called and the controller (either Ethernet of CanXL) given by CtrlIdx is referenced by a EthIfController which neither has a EthIfEthTrcvRef or EthIfCanXLTrcvRef nor a reference to a EthIfSwitch or EthIfSwitchPortGroup configured, then for this EthIfController which has a state change function configured (see EthIfTrcvLinkStateChgFunction) the User_TrcvLinkStateChg shall be called and TrcvLinkState shall be used according to the following mode / transceiver link state mapping:

1. if ETH_MODE_ACTIVE has been indicated, then ETHTRCV_LINK_STATE_ACTIVE shall be propagated as TrcvLinkState
2. if ETH_MODE_DOWN has been indicated, then ETHTRCV_LINK_STATE_DOWN shall be propagated as TrcvLinkState

⌋

[SWS_EthIf_00649] Controller mode request ETH_MODE_ACTIVE or ETH_MODE_ACTIVE_WITH_WAKEUP_REQUEST for an EthIf controller which is not referencing a transceiver, switch or switch port group

Upstream requirements: SRS_Eth_00157

⌈If EthIf_SetControllerMode has been called for an EthIfController with ETH_MODE_ACTIVE or ETH_MODE_ACTIVE_WITH_WAKEUP_REQUEST and this EthIfController has no reference to an EthIfTransceiver or EthIfSwitch or EthIfSwitchPortGroup, then EthIf shall call EthSM_TrcvLinkStateChg and all configured <User>_TrcvLinkStateChg functions with this EthIfController index and link state set to ETHTRCV_LINK_STATE_ACTIVE.⌋

[SWS_EthIf_00650] Controller mode request ETH_MODE_DOWN for an EthIf controller which is not referencing a transceiver, switch or switch port group

Upstream requirements: SRS_Eth_00157

⌈If EthIf_SetControllerMode has been called for an EthIfController with ETH_MODE_DOWN and this EthIfController has no reference to an EthIfTransceiver or EthIfSwitch or EthIfSwitchPortGroup, then EthIf shall call EthSM_TrcvLinkStateChg and all configured <User>_TrcvLinkStateChg functions with this EthIfController index and link state set to ETHTRCV_LINK_STATE_DOWN.⌋

7.1.10.3 Link state handling

The Ethernet Interface needs to supervise the link state of the maintained Ethernet hardware entities (e.g. PHY or Ethernet switch port). The link state is used by the Ethernet Interface for several tasks, e.g. forward link state changes to the upper layer. The link state is polled by the Ethernet Interface in context of a scheduled function (e.g. EthIf_MainFunctionState) by calling the according API to retrieve the link state (e.g. <EthTrcv>_GetLinkState). Thereby the Ethernet Interface need to consider the configuration:

- If an EthIfController references an EthIfTransceiver, then the link state of the corresponding Ethernet hardware entity (PHY) is supervised
- If an EthIfController references an EthIfSwitch, then the link state of the corresponding Ethernet hardware entities (i.e. Ethernet switch ports) addressed with EthIfSwitch are supervised
- If an EthIfController references an EthIfSwitchPortGroup, then the link state of the corresponding Ethernet hardware entities (i.e. Ethernet switch ports) that belong to the according EthSwtConfig are supervised

The Ethernet Interface needs to store the current link state per supervised Ethernet hardware entity (e.g. PHY or Ethernet switch port) for the internal processing.

[SWS_EthIf_00680] Supervise and store the link state per configured Ethernet hardware entity

Status: DRAFT
Upstream requirements: SRS_Eth_00033

⌈EthIf shall supervise and store the link state based on the existing configuration:

- EthIf shall supervise and store the link state per configured EthIfTransceiver
- EthIf shall supervise and store the link state of all EthSwtPorts per configured EthIfSwitch
- EthIf shall supervise and store the link state of all EthSwtPorts that are referenced by an EthIfSwitchPortGroup

⌋

Detected link state changes of supervised Ethernet hardware entities (e.g. PHYs and Ethernet switch ports) trigger several actions in the Ethernet Interface. The processing depends on the EthIfControllers configuration. As described in the introductory text of chapter Chapter 7.1.10 "Handling of maintained Ethernet hardware" one EthIfController represents one VLAN, if EthIfVlanId is configured and set to value other than '0', otherwise an EthIfController represents a software controller for so-called "untagged" traffic. The Ethernet Interface need to distinguish between the link state of its supervised Ethernet hardware entities (e.g. PHYs and Ethernet switch ports) and the link state which is reported per EthIfController towards the affected upper layers. In some cases the sole link state of a single Ethernet hardware entity needs to be reported and in some cases an accumulated link state based on the link state of multiple Ethernet hardware entities needs to be reported to the upper layer. Please refer to chapter Chapter 7.1.10.3.1 "Link state accumulation of EthIfSwitchPortGroup" for more details on the link state accumulation. The upper layer of the Ethernet Interface which are indicated regarding a link state change varies due to the configuration:

- Ethernet State Manager (EthSM) is indicated via EthSM_TrcvLinkStateChg, if the EthIfController references an EthIfTransceiver or an EthIfSwitch or an EthIfSwitchPortGroup
- Basic Software Manager (BswM) is indicated via BswM_EthIf_PortGroupLinkStateChg, for a EthIfSwitchPortGroup that is not referenced by any EthIfController
- All configured users are indicated via their individual configured API (see <User>_TrcvLinkStateChg)
- MACsec Key Agreement (MKA) is indicated via Mka_LinkStateChange, for a Ethernet hardware entity (e.g. Ethernet switch port or PHY) that is supervised and MACsec protected.

Note:

- The EthSM module uses the reported link state to control the data path of the Ethernet communication stack
- The BswM uses the reported link state to feed configured BswM rules that could perform configured actions
- The Mka uses the reported link state to process key agreement handling on a specific Ethernet hardware entity

[SWS_EthIf_00697] Either use sole link state or accumulated to indicate a link state change towards EthSM, BswM and configured users

Status: DRAFT
Upstream requirements: SRS_Eth_00033

⌈EthIf shall consider to use either the sole link state or the accumulated link state based on the following configurations to indicate a link state change to EthSM, BswM and configured users:

- If an `EthIfController` references an `EthIfTransceiver`, then the sole link state of the corresponding Ethernet hardware entity (PHY) shall be used.
- If an `EthIfController` references an `EthIfSwitch`, then the sole link state of the host port at the Ethernet switch that is connected to the corresponding Ethernet controller shall be used.
- For all `EthIfSwitchPortGroup` either referenced by an `EthIfController` or not, the accumulated link state (see [SWS_EthIf_00259]) of the `EthIfSwitchPortGroup` shall be used.

⌋

[SWS_EthIf_00682] Report of a link state change towards the EthSM and configured users for an `EthIfController` that acts independent of a MACsec operational state

Status: DRAFT
Upstream requirements: SRS_Eth_00033, FO_RS_MACsec_00002

⌈If EthIf detects a link state change either as sole link state or as accumulated link state and the following conditions are valid:

- an affected `EthIfController` references no `MkaPaeInstance`,
- this affected `EthIfController` references either an `EthIfTransceiver` or an `EthIfSwitch` or an `EthIfSwitchPortGroup`

then EthIf shall store the link state for this EthIfController and report the link state change by calling the following APIs with the changed link state (either `ETHTRCV_LINK_STATE_ACTIVE` or `ETHTRCV_LINK_STATE_DOWN`):

- call `EthSM_TrcvLinkStateChg` with `CtrlIdx` set to the affected `EthIfController` and `TransceiverLinkState` set to changed link state
- call `<User>_TrcvLinkStateChg` for all configured users with `CtrlIdx` set to the affected `EthIfController` and `TransceiverLinkState` set to changed link state

⌋

Link state changes that refer to MACsec protected Ethernet hardware entities (either PHY or Ethernet switch port) are always immediately reported to the key agreement

module to support fast processing (e.g. fast start up for MACsec protected communication).

[SWS_EthIf_00683] Always use the sole link state of a MACsec protected Ethernet hardware entities to report a link state change towards Mka

Status: DRAFT
Upstream requirements: SRS_Eth_00033

⌈EthIf shall always use the sole link state of an MACsec protected Ethernet hardware entity (either PHY or Ethernet switch port) to report a link state change towards Mka.⌋

[SWS_EthIf_00684] Report of an detected link state change for MACsec protected Ethernet hardware entities towards Mka

Status: DRAFT
Upstream requirements: SRS_Eth_00033, FO_RS_MACsec_00002

⌈If EthIf detects a link state change of an supervised Ethernet hardware entity (e.g. PHY or Ethernet switch port) where MACsec protection is enabled (e.g. `EthTrcvMacSecEnabled` set to `TRUE`) and this Ethernet hardware entity is associated with a `MkaPaeInstance` that is identified by either the following configurations:

- an `EthIfController` is referenced by an `MkaPaeInstance` and this `EthIfController` references an `EthIfTransceiver` that corresponds to the PHY where the link state change was detected
- an `EthSwtPort` is referenced by an `MkaPaeInstance` and this `EthSwtPort` corresponds to the Ethernet switch port where the link state change was detected

then EthIf shall call `Mka_LinkStateChange` with `MkaPaeIdx` set to the referencing `MkaPaeInstance` and `TransceiverLinkState` set to the sole link state (either `ETHTRCV_LINK_STATE_ACTIVE` or `ETHTRCV_LINK_STATE_DOWN`)⌋

The reporting of link state changes for an `EthIfController` towards EthSM, BswM and configured users which references a `MkaPaeInstance` are depend on the MACsec operational state of the MACsec protected Ethernet hardware entity (e.g. PHY or Ethernet switch port) where this `EthIfController` belongs to. All `EthIfController` that refer to the same `MkaPaeInstance` share this MACsec operation state (see [SWS_EthIf_00688] and [SWS_EthIf_00689]).

[SWS_EthIf_00685] Report of a link state change towards the EthSM and configured users for an `EthIfController` that depends on a MACsec operational state and references an `EthIfTransceiver`

Status: DRAFT
Upstream requirements: SRS_Eth_00033, FO_RS_MACsec_00002

⌈If the following conditions for an `EthIfController` are valid:

- an affected `EthIfController` references a `MkaPaeInstance`,
- this affected `EthIfController` references an `EthIfTransceiver`

then EthIf shall store and report a link state change towards the EthSM and configured users for this EthIfController by considering the latest available sole link state and MACsec operational state that is available for this EthIfController:

- if latest available link state is ETHTRCV_LINK_STATE_ACTIVE and the latest available MACsec operational state is set to TRUE, and the latest reported link state was ETHTRCV_LINK_STATE_DOWN, then EthIf shall report a link state change:
  - call EthSM_TrcvLinkStateChg with CtrlIdx set to this EthIfController and TransceiverLinkState set to changed link state ETHTRCV_LINK_STATE_ACTIVE
  - call <User>_TrcvLinkStateChg for all configured users with CtrlIdx set to this EthIfController and TransceiverLinkState set to ETHTRCV_LINK_STATE_ACTIVE
- in all other cases, EthIf shall report a link state change ETHTRCV_LINK_STATE_DOWN, if the last reported link state change was ETHTRCV_LINK_STATE_ACTIVE

⌋

[SWS_EthIf_00686] Report of a link state change towards the EthSM and configured users for an EthIfController that references an EthIfSwitchPortGroup

Status: DRAFT
Upstream requirements: SRS_Eth_00033, FO_RS_MACsec_00002

⌈If the following conditions for an EthIfController are valid:

- an affected EthIfController references a MkaPaeInstance,
- this affected EthIfController references an EthIfSwitchPortGroup

then EthIf shall store and report a link state change towards the EthSM and configured users for this EthIfController by considering the latest available accumulated link state (refer to [SWS_EthIf_00259] for link state accumulation):

- if latest available accumulated link state is ETHTRCV_LINK_STATE_ACTIVE and the latest reported accumulated link state was ETHTRCV_LINK_STATE_DOWN, then EthIf shall report a link state change:
  - call EthSM_TrcvLinkStateChg with CtrlIdx set to this EthIfController and TransceiverLinkState set to changed link state ETHTRCV_LINK_STATE_ACTIVE
  - call <User>_TrcvLinkStateChg for all configured users with CtrlIdx set to this EthIfController and TransceiverLinkState set to ETHTRCV_LINK_STATE_ACTIVE

- in all other cases, EthIf shall report a link state change ETHTRCV_LINK_STATE_DOWN, if the last reported accumulated link state change was ETHTRCV_LINK_STATE_ACTIVE

⌋

Note: The link state accumulation for an EthIfSwitchPortGroup take the MACsec operational state into account (see [SWS_EthIf_00687])

[SWS_EthIf_CONSTR_00011] Exclude configurations where EthIfController reference an EthIfSwitch and a MkaPaeInstance

Status: DRAFT

⌈A configuration where an EthIfController references an EthIfSwitch and a MkaPaeInstance shall be rejected as invalid.⌋

Note: AUTOSAR exclude the use case where the host port and the connected ECU that manage the Ethernet switch are MACsec protected. The EthSM and all configured users are served with the sole link state of the host port, if the corresponding EthIfController references an EthIfSwitch

[SWS_EthIf_00261] Link state reporting for EthIfSwitchPortGroup which are not referenced by any EthIfController ⌈In case an EthIfSwitchPortGroup is not connected to any EthIfController, then EthIf shall indicate the accumulated link state of the EthIfSwitchPortGroup to the BswM by calling BswM_EthIf_PortGroupLinkStateChg for the EthIfSwitchPortGroup when the link state changes (refer to [SWS_EthIf_00259] for link state accumulation).⌋

Note: Reporting of BswM_EthIf_PortGroupLinkStateChg is intentionally reporting the accumulated link state of the port group independent of any EthIf controller mode.

7.1.10.3.1 Link state accumulation of EthIfSwitchPortGroup

The Ethernet Interface needs to supervise the current link state of the EthIfSwitchPortGroups. The link state for an EthIfSwitchPortGroup is computed over all link states of the EthSwtPorts which are referenced by the EthIfSwitchPortGroup. The execution of the computation is called "link state accumulation" and the result is called "accumulated link state". The accumulated link state of the EthIfSwitchPortGroup represents the current link state of the EthIfSwitchPortGroup. The current accumulated link state of EthIfSwitchPortGroups is used to qualify the reporting towards the relevant upper layers. Please refer to the description at chapter Chapter 7.1.10.3 “Link state handling” for more details. The link state accumulation uses the link state of each EthSwtPort. Thereby EthIf needs to distinguish between the link state of MACsec protected EthSwtPorts and unprotected EthSwtPorts.

[SWS_EthIf_00687] Derivation of the `EthSwtPort` link state used for the link state accumulation

Status: DRAFT
Upstream requirements: SRS_Eth_00033, FO_RS_MACsec_00002

⌈EthIf shall consider the following rules for the link state derivation per `EthSwtPort` as prerequisite for the link state accumulation:

- If an `EthSwtPort` belongs to an `EthIfSwitchPortGroup` which is referenced by an `EthIfController` that references a `MkaPaeInstance` that protects this `EthSwtPort`, then the link state of this `EthSwtPort` shall be considered as `ETHTRCV_LINK_STATE_ACTIVE` if the current link state of the corresponding Ethernet switch port is identified as `ETHTRCV_LINK_STATE_ACTIVE` and the current MACsec operational state of this `EthSwtPort` is set to `TRUE`. Otherwise the link state of this `EthSwtPort` shall be considered as `ETHTRCV_LINK_STATE_DOWN`
- If a `EthSwtPort` belongs to a `EthIfSwitchPortGroup` which is referenced by an `EthIfController` that do not reference any `MkaPaeInstance`, then the link state of this `EthSwtPort` shall be considered as the current link state of the corresponding Ethernet switch port

⌋

[SWS_EthIf_00259] Link state accumulation of an `EthIfSwitchPortGroup` ⌈The link state for an `EthIfSwitchPortGroup` shall be computed over all link states of the `EthSwtPort`s which are referenced by the `EthIfSwitchPortGroup`. The link state of an `EthSwtPort` shall be derived with respect to [SWS_EthIf_00687]. The link state of an `EthIfSwitchPortGroup` shall result in `ETHTRCV_LINK_STATE_DOWN` if one of the following conditions is fulfilled:

- The referenced `EthSwtPort` with the role "host port" or the role "up link port" has a derived link state of `ETHTRCV_LINK_STATE_DOWN`
- Each referenced `EthSwtPort` without a role has a derived link state of `ETHTRCV_LINK_STATE_DOWN`

Otherwise the accumulated link state of an `EthIfSwitchPortGroup` shall result in `ETHTRCV_LINK_STATE_ACTIVE`.⌋

7.1.10.4 Additional Ethernet switch port handling

The following additional Ethernet switch port handling has been introduced to support a use case for a passive wake up of an ECU where all Ethernet switch ports of the corresponding Ethernet switches shall be switched on immediately. E.g. after a wakeup occurred. Afterwards it is checked if a PN request is received via NM frames within EthIfPortStartupActiveTime. If a PN request is received, then the corresponding EthIfSwitchPortGroups are requested with `ETH_MODE_ACTIVE` and corresponding Ethernet switch ports stay active. All Ethernet switch ports where the corresponding

EthIfSwitchPortGroups are not requested (due to no according PN request received within EthIfPortStartupActiveTime) are switched off.

[SWS_EthIf_00275] ⌈If `EthIf_StartAllPorts` has been called, then EthIf shall forward the call to the following functions in the given order to all EthSwtPorts of all configured EthIfSwitches:

1. Call `EthSwt_SetSwitchPortMode` with `ETH_MODE_ACTIVE`, if the current mode is `ETH_MODE_DOWN`.
2. Call `EthSwt_PortLinkStateRequest` with `ETHTRCV_LINK_STATE_ACTIVE`, if the current link state is `ETHTRCV_LINK_STATE_DOWN`

and start a timer with EthIfPortStartupActiveTime for all these ports.⌋

[SWS_EthIf_00276] ⌈After `EthIf_StartAllPorts` has been called, EthIf shall deactivate all those ports activated due to `EthIf_StartAllPorts` (see [SWS_EthIf_00275]) which are not requested with `ETH_MODE_ACTIVE` within EthIfPortStartupActiveTime by calling the following functions in the given order:

1. `EthSwt_PortLinkStateRequest` with `ETHTRCV_LINK_STATE_DOWN`
2. `EthSwt_SetSwitchPortMode` with `ETH_MODE_DOWN`

⌋

Rational: Delaying with EthIfPortStartTime is needed to ensure that NM messages with PNC information are received and the requested PNCs are activated.

Note:

1. `EthIf_StartAllPorts` could be called in context of `BswM_EcuM_CurrentWakeup`. After a wakeup occurred on the wakeup line, all EthIfSwitchPortgroups shall be activated to enable communication stack to receive NM messages (PNC information). With this it is possible to start the EthIfSwitchPortGroups without starting a PNC.
2. Further requirements for switching of EthSwtPorts, if an EthIfController referencing an EthIfSwitch are available in chapter 7.1.10.2.

7.1.11 Communication control

The Ethernet Interface has to provide a kind of communication control to support the so-called "silent communication". Silent communication is used for mode management to support a communication mode where the transmission path for a particular EthIfController is disabled, while the reception path is still enabled (see COMM_SILENT_COMMUNICATION). Disabling of the transmission path is exclusively introduced in the Ethernet Interface and has no impact on the used Ethernet hardware.

[SWS_EthIf_00504]

Upstream requirements: SRS_Eth_00157

⌈If EthIf_SetControllerMode is called for an EthIfController with ETH_MODE_ACTIVE_TX_OFFLINE and the latest accepted controller mode for this EthIfController is ETH_MODE_ACTIVE or ETH_MODE_ACTIVE_WITH_WAKEUP_REQUEST, then ETH_MODE_ACTIVE_TX_OFFLINE shall be stored as current controller mode. Otherwise the requested controller mode shall be rejected and function shall return with E_NOT_OK.⌋

Note: The transmission related API (see [SWS_EthIf_00075]) will only forward transmission requests, if the stored communication mode is ETH_MODE_ACTIVE or ETH_MODE_ACTIVE_WITH_WAKEUP_REQUEST.

7.1.12 Communication

The Ethernet Interface support a PDU-based communication approach to transfer data from the lower layer to the upper layer and vice versa. The EthIf module interchange PDUs with the upper layers (e.g. [17, SWS IEEE1722Tp module]) via the [18, SWS L-SDU router module]. This approach interchanges frame-specific information via specific PDUs. The APIs carries PduId and PduInfoPtr. PduId identifies the according PDU. PduInfoPtr contains SduDataPtr, which addresses the location of buffer where data is provided and SduDataLength which denotes the length of the provided data. Optionally, meta data can be used to transfer additional frame specific information (see 7.1.12.5 “Meta data handling”)

PDUs are configured in PDU pools. Multiple PDU pools could be configured per EthIfFrameType. The PDU pools are organized in EthIfFrameRxPools and EthIfFrameTxPools. A EthIfFrameRxPool could contain multiple EthIfFrameRxPdus and a EthIfFrameTxPool multiple EthIfFrameTxPdus. A PDU pool could reference a specific EthIfController, i.e. the PDU pool is utilized for communication via this specific EthIfController. Such PDU pools are called "fixed" PDU pools. Fixed PDU pools are used per EthIfFrameType and EthIfController. PDU pools without an configured reference to an specific EthIfController are called "floating" PDU pools. Such PDU pools are configured per EthIfFrameType and shared across configured EthIfController. PDUs of PDU pools should have the same PDU properties configured, since they are used for the same purpose.

[SWS_EthIf_CONSTR_00010] Same configuration of PDUs that belong to the same PDU pool for `KeepLocalPduBuffer`

Status: DRAFT

⌈All EthIfFrameRxPdus that belong to the same EthIfFrameRxPool and all EthIfFrameTxPdus that belong to same EthIfFrameTxPool shall have the same value for KeepLocalPduBuffer configured (either TRUE or FALSE)⌋

Note: Please refer to (see 7.1.12.5 “Meta data handling”) for configuration of MetaDataItems

The EthIf need to translate between PDU-based communicaton and frame-based communication for the interaction with the Eth driver and vice versa, since the Eth driver is not PDU-aware:

- The Eth driver provide frame specific API parameter via EthIf_RxIndication and EthIf translate the frame specific parameter to a PDU-based reception indication.
- The upper layer modules (e.g. IEEE1722Tp module) request the EthIf to transmit PDUs with frame specific information via meta data (e.g. VLAN-priority), and the EthIf translate the PDU-based transmission request to an frame-based transmission request towards the Eth driver:
  - direct data provision: the upper layer request EthIf to forward the provided data directly to Eth driver
  - indirect data provision: the upper layer request EthIf to call the upper layers TriggerTransmit function to retrieve data, and EthIf trigger transmission afterwards

7.1.12.1 Reception

If EthIf is indicated via a call of EthIf_RxIndication to receive an Ethernet frame, then EthIf need to check if the frame type of the received Ethernet frame matches to a configured EthIfFrameType. If a match is identified and the referencing EthIfFrameConfig of the matching EthIfFrameType have an EthIfFrameRxPool configured, then EthIf has to search for an available EthIfFrameRxPdu. If a EthIfFrameRxPdu is available, then this EthIfFrameRxPdu is used to perform the receive processing.

The reception is indicated and forwarded to the destination upper layer. The destination upper layer receive an indication and perform a reception processing. Optionally, if the reception processing is finalized, the upper layer could explicitly indicate to release the EthIfFrameRxPdu with a call of EthIf_ReleaseRxBuffer. This behaviour is configurable and used to support hardware supported data transfer (e.g. via DMA) from the lower layer buffers to the upper layers destination receive buffer. As long as the asynchronous data transfer is not finalized, the EthCtrlConfigIngressQueue element is locked, and consequently also the used EthIfFrameRxPdu. For further receptions that matches the same EthIfFrameType another EthIfFrameRxPdu of the corresponding EthIfFrameRxPool has to be used. A call of EthIf_ReleaseRxBuffer is only expected by the EthIf, if a global PDU is configured with KeepLocalPduBuffer set to TRUE. KeepLocalPduBuffer set to TRUE indicate, that the destination upper layer may trigger a hardware supported data transfer. Therefore the EthCtrlConfigIngressQueue element need to be locked until the upper layer indicate to release EthCtrlConfigIngressQueue element. If a global PDU is configured with

KeepLocalPduBuffer set to FALSE, the EthIf module call directly <EthDrv>_ReleaseRxBuffer after the RxIndication function call returns.

[SWS_EthIf_00663] Reception handling with fixed EthIfFrameRxPools

Status: DRAFT
Upstream requirements: SRS_BSW_00350, SRS_BSW_00386

⌈If EthIf_RxIndication is called and the following conditions are true:

- the given FrameType match to EthIfFrameType of a configured EthIfFrameConfig
- the matching EthIfFrameConfig has an EthIfFrameRxPool configured, where at least one EthIfFrameRxPdu is in state PDU_AVAILABLE
- the VLAN-ID of the received Ethernet frame match to the EthIfVlanId of the EthIfController which is referenced via EthIfFrameRxControllerRef

then the EthIf shall perform the following actions:

- set the RxPduId to the PDU-ID of the PDU, which is referenced by the EthIfFrameRxPdu
- transfer the given DataPtr and DataLen to the PduInfoPtr of the used EthIfFrameRxPdu
- if MetaDataItem TIMETUPLE_TYPE_PTR is configured at the used PDU, which is referenced by the EthIfFrameRxPdu:
  - produce MetaDataItem TIMETUPLE_TYPE_PTR and transfer the given IngressTimeTuplePtr to the produced MetaDataItem TIMETUPLE_TYPE_PTR
  - add a pointer of the produced MetaDataItem to the PduInfoPtr of the used EthIfFrameRxPdu
- set the EthIfFrameRxPdu to state PDU_IN_USE
- store a mapping of the given RxHandleId with the PDU-ID of the used EthIfFrameRxPdu
- call LSduR_EthIfRxIndication with created PduInfoPtr and RxPduId of the used EthIfFrameRxPdu

⌋

[SWS_EthIf_00664] Reception handling with floating EthIfFrameRxPools

Status: DRAFT
Upstream requirements: SRS_BSW_00350, SRS_BSW_00386

⌈If EthIf_RxIndication is called and the following conditions are true:

- the given FrameType match to EthIfFrameType of a configured EthIfFrameConfig

- the matching EthIfFrameConfig has an EthIfFrameRxPool configured, where at least one EthIfFrameRxPdu is in state PDU_AVAILABLE
- the EthIfFrameRxPool is not assigned to an specific EthIfController, i.e. EthIfFrameRxControllerRef is not configured
- the VLAN-ID of the received Ethernet frame match to the EthIfVlanId of an configured EthIfController

then the EthIf shall perform the following actions:

- set the RxPduId to the PDU-ID of the PDU, which is referenced by the EthIfFrameRxPdu
- transfer the given DataPtr and DataLen to the PduInfoPtr of the used EthIfFrameRxPdu
- if MetaDataItem TIMETUPLE_TYPE_PTR is configured at the used PDU, which is referenced by the EthIfFrameRxPdu:
  - produce MetaDataItem TIMETUPLE_TYPE_PTR and transfer the given IngressTimeTuplePtr to the produced MetaDataItem TIMETUPLE_TYPE_PTR
  - add a pointer of the produced MetaDataItem to the PduInfoPtr of the used EthIfFrameRxPdu
- set the EthIfFrameRxPdu to state PDU_IN_USE
- store a mapping of the given RxHandleId with the PDU-ID of the used EthIfFrameRxPdu
- call LSduR_EthIfRxIndication with created PduInfoPtr and RxPduId of the used EthIfFrameRxPdu

⌋

[SWS_EthIf_00665] Abort of reception indication process

Status: DRAFT
Upstream requirements: SRS_BSW_00350, SRS_BSW_00386

⌈If EthIf_RxIndication is called, then the EthIf shall first try to find a matching EthIfFrameRxPdu from an fixed EthIfFrameRxPool and second from an floating EthIfFrameRxPool. If no matching EthIfFrameRxPdu is available, then the processing of the reception indication shall be aborted.⌋

[SWS_EthIf_00638] Error handling for aborted reception indication process

Status: DRAFT
Upstream requirements: SRS_BSW_00350, SRS_BSW_00386

⌈If a reception indication processing is aborted, then EthIf shall call <EthDrv>_ReleaseRxBuffer with the given RxHandleId.⌋

[SWS_EthIf_00639] Recpetion handling for PDUs with `KeepLocalPduBuffer` set to `FALSE`

Status: DRAFT
Upstream requirements: SRS_BSW_00350, SRS_BSW_00386

⌈If `LSduR_EthIfRxIndication` returns and the used PDU-ID refer to a global PDU which has `KeepLocalPduBuffer` set to `FALSE`, then the EthIf shall perform the following actions:

- set the affected `EthIfFrameRxPdu` to state `PDU_AVAILABLE`
- release the local buffer produced for this PDU
- remove the mapping between the `RxHandleId` and the PDU-ID of the used `EthIfFrameRxPdu`
- call `<EthDrv>_ReleaseRxBuffer` with `RxHandleId` mapped to the given `RxPduId`

⌋

[SWS_EthIf_00640] Recpetion handling for PDUs with `KeepLocalPduBuffer` set to `TRUE`

Status: DRAFT
Upstream requirements: SRS_BSW_00350, SRS_BSW_00386

⌈If `LSduR_EthIfRxIndication` returns and the used PDU-ID refer to a global PDU which has `KeepLocalPduBuffer` set to `TRUE`, then the EthIf shall keep the local buffer and the state of the used PDU, until `EthIf_ReleaseRxBuffer` for the used PDU-ID is called.⌋

[SWS_EthIf_00641] Handling if `EthIf_ReleaseRxBuffer` is called

Status: DRAFT
Upstream requirements: SRS_BSW_00350, SRS_BSW_00386

⌈If `EthIf_ReleaseRxBuffer` is called and the following conditions are true:

- the given `PduId` refer to a PDU-ID, where a mapping is stored to a corresponding `RxHandleId`
- the `EthIfFrameRxPdu` is in state `PDU_IN_USE`

then the EthIf shall perform the following actions. Otherwise the function shall return without doing any action:

- set the affected `EthIfFrameRxPdu` to state `PDU_AVAILABLE`
- release the local buffer produced for this PDU
- remove the mapping between the `RxHandleId` and the PDU-ID of the used `EthIfFrameRxPdu`
- call `<EthDrv>_ReleaseRxBuffer` with `RxHandleId` mapped to the given `RxPduId`

⌋

Note:

- EthIf_ReleaseRxBuffer could be called by the upper layer in context of the LSduR_EthIfRxIndication
- <EthDrv>_ReleaseRxBuffer could be called by the EthIf in context of the EthIf_RxIndication

7.1.12.2 Transmission

If EthIf is requested via a call of EthIf_Transmit to transmit a PDU, then EthIf need to check the availability of EthIfFrameTxPdu addressed with the given TxPduId. For an available EthIfFrameTxPdu, EthIf transform the given PduInfoPtr together with the given meta data to a frame-based API call towards the Eth driver. Whereat, the EthIf module need to consider the data provision of the upper layer. The EthIf module support transmission requests with "indirect data provision" and with "direct data provision":

- Indirect data provision: If EthIf_Transmit is called with PduInfoPtr.SduDataPtr set to NULL_PTR, then the EthIf allocate a queue element at an egress queue with a call to <EthDrv>_ProvideTxBuffer, call the TriggerTransmit function from the upper layer via LSduR_EthIfTriggerTransmit and call afterwards <EthDrv>_Transmit to trigger the transmission of the data stored in the allocated egress queue element.
- direct data provision: If EthIf_Transmit is called with PduInfoPtr.SduDataPtr set to data pointer, then the EthIf prepare the data and forward the call directly to the Eth driver via a call of <EthDrv>_ImmediateTransmit

[SWS_EthIf_00670] Evaluation of transmission request with direct data provision

Status: DRAFT
Upstream requirements: SRS_BSW_00350, SRS_BSW_00386

⌈If EthIf_Transmit is called with PduInfoPtr.SduDataPtr set to a data pointer, then the EthIf shall proceed with an deferred forwarding, if the following conditions are true. Otherwise the EthIf shall proceed with an immediate forwarding:

- the given TxPduId match to a EthIfFrameTxPduId aggregated by a EthIfFrameTxPdu of a configured EthIfFrameTxPool
- the EthIfFrameTxPdus of this EthIfFrameTxPool have KeepLocalPduBuffer set to FALSE
- the affected EthIfController refer to an EthIfPhysController that references an EthCtrlConfig at the Eth driver that have EthCtrlEnableEgressHardwareSupportedDataTransfer set to TRUE

⌋

[SWS_EthIf_00666] Transmission request with direct data provision and immediate forwarding

Status: DRAFT
Upstream requirements: SRS_BSW_00350, SRS_BSW_00386

⌈If EthIf qualified an immediate forwarding (see [SWS_EthIf_00670]) for a transmission request with direct data provision and the following conditions are true:

- the given `TxPduId` match to a `EthIfFrameTxPduId` aggregated by a `EthIfFrameTxPdu` of a configured `EthIfFrameTxPool`
- the referenced PDU of the matching `EthIfFrameTxPduId` is in state `PDU_AVAILABLE`

then the EthIf shall perform the following actions as preparation for a call of `<EthDrv>_ImmediateTransmit`. Otherwise the function shall return with `E_NOT_OK`:

- set the `CtrlIdx` parameter to the corresponding `EthIfPhysController` which belongs to the referencing `EthIfController`
- set the `TxHandleId` parameter to the matching `EthIfFrameTxPduId` of the used `EthIfFrameTxPdu`
- set `priority` to configured value of `EthIfFrameTxPriority` of the corresponding `EthIfFrameTxPool`, if available, otherwise to the priority from the configured `MetaDataItem PRIORITY_8`. If neither `EthIfFrameTxPriority` nor `MetaDataItem PRIORITY_8` is available, then set priority to 0.
- create a list-element-struct of type `ListElemStructType` according to [SWS_EthIf_00667] and set the `HeaderListPtr` parameter to the address of the created list-element-struct
- set the `PayloadPtr` to the `SduDataPtr` given with the received `PduInfoPtr` and the `PayloadLength` to the `SduLength` given with the received `PduInfoPtr`
- call `<EthDrv>_ImmediateTransmit` with `CtrlIdx`, `TxHandleId`, `priority`, `HeaderListPtr`, `PayloadPtr` and `PayloadLength` set to values as described by the previous steps

⌋

[SWS_EthIf_00667] Creation of a list-element-struct of type `ListElemStructType`

Status: DRAFT
Upstream requirements: SRS_BSW_00350, SRS_BSW_00386

⌈If EthIf has to create a list-element-struct of type `ListElemStructType` due to a transmission request with direct data provision and the given `TxPduId` match to `EthIfFrameTxPduId` which reference a PDU that has `MetaDataItemType`

ETHERNET_MAC_64 configured, then EthIf shall consider the following points to create a list-element-struct of type ListElemStructType:

- consume destination MAC address from the MetaDataItem ETHERNET_MAC_64
- derive the source MAC address from the EthCtrl which corresponds to the EthIfPhysController that is referenced by the EthIfController that belongs to the respective EthIfFrameTxPool
- use the EthIfFrameType of the EthIfFrameConfig which aggregates the EthIfFrameTxPool that refers to the used EthIfFrameTxPdu
- derive the VLAN-ID of the EthIfController which is referenced by the EthIfFrameTxPool of the used EthIfFrameTxPdu
- use the configured priority value of EthIfFrameTxPriority of EthIfFrameTxPool which aggregates the used EthIfFrameTxPdu, if available, otherwise the priority of the configured MetaDataItem PRIORITY_8. If neither EthIfFrameTxPriority nor MetaDataItem PRIORITY_8 is available, then set priority to 0.
- create an Ethernet header according the [19, IEEE 802.3 Std 2022] (DstMacAdr;SrcMacAdr;QTag;EtherType) and use the pointer to the constructed header for DataPtr and length of the Ethernet header for DataLength of the create a list-element-struct
- If the referenced PDU has MetaDataItemType LISTELEM_PTR configured, then consume the list element pointer from MetaDataItem LISTELEM_PTR and set NextListElemPtr of the created list-element-struct to the address of the consumed list element pointer. Otherwise set NextListElemPtr of the created list-element-struct to NULL_PTR.
- If the referenced PDU has MetaDataItemType VLAN_16 configured, then consume the VLAN ID from this MetaDataItem to identify the right EthIfController based on the configured EthIfPhysController.

⌋

[SWS_EthIf_00671] Transmission request with direct data provision and deferred forwarding

Status: DRAFT
Upstream requirements: SRS_BSW_00350, SRS_BSW_00386

⌈If EthIf qualified a deferred forwarding (see [SWS_EthIf_00670]) for a transmission request with direct data provision and the following conditions are true:

- the given TxPduId match to a EthIfFrameTxPduId aggregated by a EthIfFrameTxPdu of a configured EthIfFrameTxPool
- the referenced PDU of the matching EthIfFrameTxPduId is in state PDU_AVAILABLE

then the EthIf shall perform the following actions. Otherwise the function shall return with E_NOT_OK:

- mark the transmission request as direct data provision with deferred forwarding
- proceed with preparation for a call of <EthDrv>_ProvideTxBuffer according to [SWS_EthIf_00672]

⌋

[SWS_EthIf_00677] Return of <EthDrv>_ProvideTxBuffer for transmission request with direct data provision and deferred forwarding

Status: DRAFT
Upstream requirements: SRS_BSW_00350, SRS_BSW_00386

⌈If <EthDrv>_ProvideTxBuffer returns with E_OK for transmission request with direct data provision and deferred forwarding, then EthIf shall perform the following actions, otherwise release all resources prepared for this call (see [SWS_EthIf_00672]) and return the EthIf_Transmit call with E_NOT_OK:

- if affected EthIfController has EthIfVlanId configured, then store the following frame attributes at the beginning of the provided BufPtr: priority (VLAN-priority) determined within the preparation for call of <EthDrv>_ProvideTxBuffer (see [SWS_EthIf_00672]), CFI (always 0), VID (configured EthIfVlanId) and the configured EthIfFrameType associated with EthIfFrameTxPdu
- copy data provided via PduInfoPtr.SduDataPtr with PduInfoPtr.SduDataLength to next position after the previous added frame attributes
- prepare a call of <EthDrv>_Transmit:
  - set the CtrlIdx parameter to the corresponding EthIfPhysController which belongs to the referencing EthIfController of the corresponding EthIfFrameTxPdu
  - set BufIdx which is assigned to this transmission request
  - set FrameType to configured EthIfFrameType associated with EthIfFrameTxPdu
  - set TxConfirmation to TRUE
  - set LenByte to total length of bytes written to the provided buffer addressed via BufPtr
  - set PhysAddrPtr to location where the MAC destination address associated with EthIfFrameTxPdu (see [SWS_EthIf_00672]) is available
  - call <EthDrv>_Transmit with CtrlIdx, BufIdx, FrameType, TxConfirmation, LenByte, PhysAddrPtr with values determined by previous steps

⌋

[SWS_EthIf_00678] `<EthDrv>_Transmit` return `E_NOT_OK` for transmission request with direct data provision and deferred forwarding

Status: DRAFT
Upstream requirements: SRS_BSW_00350, SRS_BSW_00386

⌈If `<EthDrv>_Transmit` returns with `E_NOT_OK` from a transmission request with direct data provision and deferred forwarding, then EthIf shall return the `EthIf_Transmit` with `E_NOT_OK`⌋

[SWS_EthIf_00676] Transmission request with indirect data provision

Status: DRAFT
Upstream requirements: SRS_BSW_00350, SRS_BSW_00386

⌈If `EthIf_Transmit` is called with `PduInfoPtr.SduDataPtr` set to `NULL_PTR` and the following conditions are true:

- the given `TxPduId` match to a `EthIfFrameTxPduId` aggregated by a `EthIfFrameTxPdu` of a configured `EthIfFrameTxPool`
- the referenced PDU of the matching `EthIfFrameTxPduId` is in state `PDU_AVAILABLE`

then the EthIf shall perform the following actions. Otherwise the function shall return with `E_NOT_OK`:

- mark the transmission request as indirect data provision
- proceed with preparation for a call of `<EthDrv>_ProvideTxBuffer` according to [SWS_EthIf_00672]

⌋

[SWS_EthIf_00672] Preparation for call `<EthDrv>_ProvideTxBuffer`

Status: DRAFT
Upstream requirements: SRS_BSW_00350, SRS_BSW_00386

⌈If EthIf is requested to perform a preparation for a call of `<EthDrv>_ProvideTxBuffer`, the following points shall be considered:

- set the `CtrlIdx` parameter to the corresponding `EthIfPhysController` which belongs to the referencing `EthIfController`
- set `priority` to configured value of `EthIfFrameTxPriority` of the corresponding `EthIfFrameTxPool`, if available, otherwise to the priority from the configured `MetaDataItem` `PRIORITY_8`. If neither `EthIfFrameTxPriority` nor `MetaDataItem` `PRIORITY_8` is available, then set priority to 0.
- allocate buffer of type `Eth_BufIdxType` and set the `BufIdxPtr` to the allocated buffer
- consume `MetaDataItem`, if configured:

  - destination MAC address from the MetaDataItem ETHERNET_MAC_64
  - time tuple from the MetaDataItem TIMETUPLE_TYPE_PTR
- allocate buffer for out-parameter BufPtr
- allocate buffer for the inout-paramter LenBytePtr and set the value to PduInfoPtr.SduDataLength, if the affected EthIfController has no EthIfVlanId configured or EthIfVlanId is set 0. Otherwise set the value to PduInfoPtr.SduDataLength + 4 as desired length
- create a mapping of EthIfFrameTxPdu, allocated buffer and consumed meta data
- call Eth_ProvideTxBuffer with <EthDrv>_ProvideTxBuffer, priority, BufPtr and LenBytePtr set to values as described by the previous steps

⌋

[SWS_EthIf_00673] Return of <EthDrv>_ProvideTxBuffer for transmission request with indirect data provision

Status: DRAFT
Upstream requirements: SRS_BSW_00350, SRS_BSW_00386

⌈If <EthDrv>_ProvideTxBuffer returns with E_OK for transmission request with indirect data provision, then EthIf shall perform the following actions as preparation for a call of LSduR_EthIfTriggerTransmit, otherwise release all resources prepared for this call (see [SWS_EthIf_00672]) and return the EthIf_Transmit call with E_NOT_OK:

- if affected EthIfController has EthIfVlanId configured, then store the following frame attributes at the beginning of the provided BufPtr: priority (VLAN-priority) determined within the preparation for call of <EthDrv>_ProvideTxBuffer (see [SWS_EthIf_00672]), CFI (always 0), VID (configured EthIfVlanId) and the configured EthIfFrameType associated with EthIfFrameTxPdu
- if affected EthIfController has EthIfVlanId configured, then set PduInfoPtr.SduDataLength to value of LenBytePtr - 4, otherwise set value to PduInfoPtr.SduDataLength to LenBytePtr as output granted length
- if affected EthIfController has EthIfVlanId configured, set PduInfoPtr.SduDataPtr to BufPtr + offset of 4, otherwise set PduInfoPtr.SduDataPtr to BufPtr
- assign the egress buffer idenitifier provided via BufIdxPtr to this transmission request identified with EthIfFrameTxPduId of the corresponding EthIfFrameTxPdu
- call LSduR_EthIfTriggerTransmit with TxPduId set to the corresponding EthIfFrameTxPdu and PduInfoPtr with values of the previous steps

⌋

[SWS_EthIf_00674] Return of `LSduR_EthIfTriggerTransmit` for transmission request with indirect data provision

Status: DRAFT
Upstream requirements: SRS_BSW_00350, SRS_BSW_00386

⌈If `LSduR_EthIfTriggerTransmit` returns with `E_OK` from a transmission request with indirect data provision, then EthIf shall perform the following actions as preparation for a call of `<EthDrv>_Transmit`, otherwise release all resources prepared for this call (see [SWS_EthIf_00673]) and call `LSduR_EthIfTxConfirmation` with `TxPduId` set to the corresponding `EthIfFrameTxPdu` and `result` set `E_NOT_OK`:

- set the `CtrlIdx` parameter to the corresponding `EthIfPhysController` which belongs to the referencing `EthIfController` of the corresponding `EthIfFrameTxPdu`
- set `BufIdx` which is assigned to this transmission request (see [SWS_EthIf_00673])
- set `FrameType` to configured `EthIfFrameType` associated with `EthIfFrameTxPdu`
- set `TxConfirmation` to `TRUE`
- set `LenByte` to accumulated length returned via `PduInfoPtr.SduDataLength` and length of stored frame attributes prepared for a call of `<EthDrv>_ProvideTxBuffer` (see [SWS_EthIf_00673])
- set `PhysAddrPtr` to location where the MAC destination address associated with `EthIfFrameTxPdu` (see [SWS_EthIf_00672]) is available
- call `<EthDrv>_Transmit` with `CtrlIdx`, `BufIdx`, `FrameType`, `TxConfirmation`, `LenByte`, `PhysAddrPtr` with values determined by previous steps

⌋

[SWS_EthIf_00675] `<EthDrv>_Transmit` return `E_NOT_OK` for transmission request with indirect data provision

Status: DRAFT
Upstream requirements: SRS_BSW_00350, SRS_BSW_00386

⌈If `<EthDrv>_Transmit` returns with `E_NOT_OK` from a transmission request with indirect data provision, then EthIf shall call `LSduR_EthIfTxConfirmation` with `TxPduId` set to the corresponding `EthIfFrameTxPdu` and `result` set `E_NOT_OK`⌋

[SWS_EthIf_00600] Finalization of transmission request with direct or indirect data provision

Status: DRAFT
Upstream requirements: SRS_BSW_00350, SRS_BSW_00386

⌈If <EthDrv>_ImmediateTransmit or <EthDrv>_Transmit returns with E_OK and the used PDU-ID refer to a global PDU which has KeepLocalPduBuffer set to TRUE, then the EthIf shall keep the local buffer and the state of the used PDU, until EthIf_TxConfirmation for the used PDU-ID is called. In all other cases, where EthIf calls <EthDrv>_ImmediateTransmit or<EthDrv>_Transmit, the buffer for local produced data of the affected PDU shall be released.⌋

7.1.12.3 Transmission confirmation

[SWS_EthIf_00644]

Status: DRAFT
Upstream requirements: SRS_BSW_00350, SRS_BSW_00386

⌈If EthIf_TxConfirmation is called and the following conditions are true:

- the given BufIdx and CtrlIdx refer to a EthIfFrameTxPdu (given in previous call of <EthDrv>_ImmediateTransmit as TxHandleId or given with previous call of <EthDrv>_Transmit as BufIdx
- the affected EthIfFrameTxPdu is in state PDU_IN_USE

then the EthIf shall perform the following action, otherwise abort transmission processing and return:

- set the affected PDU of EthIfFrameTxPdu to state PDU_AVAILABLE
- call LSduR_EthIfTxConfirmation with TxPduId set to PDU-ID referenced EthIfFrameTxPdu

⌋

[SWS_EthIf_00115] Polling mode to trigger transmission confirmation ⌈In each call of EthIf_MainFunctionTx the component shall call <EthDrv>_TxConfirmation for all Ethernet Controller Drivers.⌋

Note: The Ethernet Interface expects that each Ethernet Controller Driver issues confirmations for all transmitted frames using the call-back function EthIf_TxConfirmation.

7.1.12.4 State handling of PDUs

The EthIf module has to maintain the usage-state of PDUs from the according PDU-pool. Therefore PDUs have two states PDU_IN_USE or PDU_AVAILABLE.

Note: The definition of PDU_IN_USE or PDU_AVAILABLE represents only the functional behavior, but not the implementation, since the state of a PDU is kept locally and is not propagated to other modules. Therefore, no type definition for the PDU state is specified.

[SWS_EthIf_00645]
Status: DRAFT
Upstream requirements: SRS_BSW_00350, SRS_BSW_00386

⌈The EthIf module shall maintain for each PDU of all configured EthIfFrameRxPdus and EthIfFrameTxPdus two states: state PDU_AVAILABLE and state PDU_IN_USE⌋

[SWS_EthIf_00646]
Status: DRAFT
Upstream requirements: SRS_BSW_00350, SRS_BSW_00386

⌈If the EthIf module is requested to transmit data or is indicated to receive data, or if transmission confirmation or release reception buffer is indicated, then the EthIf module shall check the state of the PDU according the given PDU-ID:

- If the PDU of the given PDU-ID is in state PDU_AVAILABLE and requested to be transmitted or indicated to be received, then the EthIf module shall set the state of this PDU to PDU_IN_USE. Otherwise the EthIf module shall abort further handling, report a runtime error ETHIF_E_PDU_STATE_TRANSITION_FAILED and, if possible return with E_NOT_OK.
- If the PDU of the given PDU-ID is in state PDU_IN_USE and transmission confirmation or release reception buffer is indicated, then the EthIf module shall set the state of this PDU to PDU_AVAILABLE. Otherwise the EthIf module shall abort further handling, report an runtime error ETHIF_E_PDU_STATE_TRANSITION_FAILED and return.

⌋

[SWS_EthIf_00647]
Status: DRAFT
Upstream requirements: SRS_BSW_00350, SRS_BSW_00386

⌈If the EthIf module is requested to transmit data and the function call <EthDrv>_ImmediateTransmit or Eth_Transmit returns with E_NOT_OK, then the IEEE1722Tp module shall set the state of the affected PDU to PDU_AVAILABLE.⌋

7.1.12.5 Meta data handling

The EthIf module uses meta data as specified in [6, CP-SWS-BSWGeneral]. Meta data are addressed with the MetaDataPtr, which is part of the PduInfoPtr. Basically, the EthIf module act as intermediate layer to transfer provided (frame-based) data from the Ethernet driver to the upper layer as PDUs, and to transfer PDUs from the upper layer communication stack to Ethernet driver as frame-related API call. In both directions

the EthIf module need to translate between the frame-based approach and the PDU-based approach and vice versa. The following communication scenarios have to be considered:

- UpperLayer-To-LowerLayer-TxData: upper layer (e.g. IEEE1722Tp) data transmission via the LSduR module to EthIf
- LowerLayer-To-UpperLayer-RxData: EthIf reception indication of Ethernet frame, transformation of data PDU-based approach and forwarding of data to upper layer (e.g. IEEE1722Tp) via LSduR

7.1.12.5.1 Meta data types

This sub chapter describe the expected meta data types, which are produces or consumed by EthIf.

[SWS_EthIf_CONSTR_00002]

Status: DRAFT

⌈A PDU which refer to an `EthIfFrameRxPdu`, shall have no other `MetaDataItem` of `MetaDataItemType` configured than:

- `TIMETUPLE_TYPE_PTR`
- `ETHERNET_MAC_64`
- `BROADCAST_8`

⌋

[SWS_EthIf_CONSTR_00003]

Status: DRAFT

⌈A PDU which refer to an `EthIfFrameTxPdu`, shall have no other `MetaDataItem` of `MetaDataItemType` configured than:

- `ETHERNET_MAC_64`
- `PRIORITY_8`
- `TIMETUPLE_TYPE_PTR`
- `LISTELEM_PTR`
- `VLAN_16`

⌋

7.1.12.6 Ingress queue handling

The Ethernet interface module support different approaches for ingress queue handling. Ingress queue handlng higly depends on the configured ingress queue processing in context of the Ethernet driver. The Ethernet driver support the following approaches:

- all ingress queues of an specific Ethernet controller are handled in interrupt mode
- all ingress queues of an specific Ethernet controller are handled in polling mode
- specific ingress queue of an specific Ethernet controller handled in interrupt mode and the remaing ingress queues in polling mode

The polling mode need to destinguish which function is reponsible for polling a specific ingress queue:

1. If an `EthIfPhysController` reference multiple ingress queues via `EthIfPhysCtrlRxMainFunctionIngressQueueProcessing`, then the referenced queues handled in a specific `EthIf_MainFunctionRx_<IngressQueueProcessing ShortName>`
2. If the corresponding EthController of an `EthIfPhysController` have ingress queues configured with an `EthCtrlConfigIngressQueueHandlerFunction`, then this ingress queue is handled within an specific ingress queue handler function. The scheduling of this function is integration specific (e.g. scheduled by an CDD or mapped to an Os task)
3. All ingress queues which have no specific ingress queue handler function configured, are handled in the context of the `EthIf_MainFunctionRx`

The Ethernet driver support to handle specific ingress queues in interrput mode A `EthIfPhysController`could configure multiple Rx mainfunctions to handle specific ingress queues by using `EthIfPhysCtrlRxMainFunctionIngressQueueProcessing`. A `EthIfPhysCtrlRxMainFunctionIngressQueueProcessing` could reference one ingress queue via `EthIfCanXLCtrlRxIngressFifoRef` or `EthIfPhysCtrlRxIngressQueueRef`. Along with this reference a specific main function is generated (see `EthIf_MainFunctionRx_<IngressQueueProcessing ShortName>`), where the ingress queue handler is implemented.

[SWS_EthIf_CONSTR_00004]

Status: DRAFT

⌈If a `EthIfPhysController` have `EthIfPhysCtrlRxMainFunctionIngressQueueProcessing` configured and reference ingress queues via `EthIfPhysCtrlRxIngressQueueRef` or `EthIfCanXLCtrlRxIngressFifoRef`, then the configuration shall be qualified as valid, if the referenced ingress queues have neither `EthCtrlConfigIngressQueueHandlerFunction` nor `EthCtrlEnableIngressQueueInterrupt` set to `TRUE` configured⌋

[SWS_EthIf_CONSTR_00005]

Status: DRAFT

⌈If a EthIfPhysController have EthIfPhysCtrlRxMainFunctionIngressQueueProcessing configured, then the referenced ingress queues via EthIfPhysCtrlRxIngressQueueRef or EthIfCanXLCtrlRxIngressFifoRef shall be handled by the corresponding EthIf_MainFunctionRx_<IngressQueueProcessing ShortName>⌋

The generic EthIf_MainFunctionRx could perform a polling for all ingress queues, which are not handled by other ingress handler functions. Other ingress queue handler functions could be provided by the EthIf or by the Eth driver. The existence of other handler functions influences the ingress queue handling of EthIf_MainFunctionRx.

[SWS_EthIf_00648]

Status: DRAFT
Upstream requirements: SRS_Eth_00170

⌈If EthIf has EthIfEnableRxInterrupt is set to FALSE, then the EthIf_MainFunctionRx shall perform the polling for ingress queues where all following conditions apply:

- an ingress queue is neither referenced by EthIfPhysCtrlRxIngressQueueRef nor by EthIfCanXLCtrlRxIngressFifoRef
- an ingress queue at the Ethernet driver has EthCtrlEnableIngressQueueInterrupt set to FALSE
- an ingress queue at the Ethernet driver has no EthCtrlConfigIngressQueueHandlerFunction configured

⌋

7.1.13 Global Time support

For more details regarding time measurement with Switches, please refer to [20, Specification of Ethernet Switch Driver].

7.1.14 Wireless Ethernet Support

[SWS_EthIf_00340] ⌈The Ethernet Interface shall support Wireless Ethernet specific functionality, depending on the parameter EthIfEnableWEthApi.⌋

The Wireless functions are divided in controller and transceiver specific functionality. Mainly, transmission and reception parameters are being exchanged with the EthIf upper module and the controller/transceiver.

The controller is being called only for buffer specific transmission and reception parameters by the APIs:

- EthIf_GetBufWRxParams
- EthIf_GetBufWTxParams
- EthIf_SetBufWTxParams

The Transceiver is being called for general configuration of the wireless radio and the wireless radio's channel by:

- EthIf_SetRadioParams
- EthIf_SetChanRxParams
- EthIf_SetChanTxParams
- EthIf_GetChanRxParams

The parameter values are requested or transmitted by unique parameter identifiers. They are defined within the controller and transceiver specification [16] [21].

7.1.15 Cellular V2X Support

[SWS_EthIf_00520]

Status: DRAFT

⌈The Ethernet Interface shall support Cellular V2X specific functionality, depending on the parameter EthIfEnableCV2xApi⌋

Transmission and reception parameters are being exchanged with the EthIf upper module and the controller. The controller is being called only for buffer specific transmission and reception parameters by the APIs:

- EthIf_GetBufCV2xPC5RxParams
- EthIf_GetBufCV2xPC5TxParams
- EthIf_SetBufCV2xPC5TxParams

The controller is being called for general configuration of the Cellular V2X radio and the Cellular V2X radio's channel by:

- EthIf_GetChanCV2xPC5TxParams

The parameter values are requested or transmitted by unique parameter identifiers. They are defined within the controller specification [22].

7.1.16 MACsec support

AUTOSAR supports MACsec in hardware hosted by an Ethernet controller, an Ethernet switch port or a PHY. EthIf, in its role as a hardware abstraction towards the upper layer modules, needs to check the MACsec configuration across its maintained hardware-dependent drivers for consistency.

Please note: MACsec support in hardware in context of AUTOSAR needs to respect the constraints on the hardware accessibility via one type of management interface (e.g. MDIO). Please refer to Chapter 4.1 “Limitations” for more details.

[SWS_EthIf_CONSTR_00012] MACsec configuration consistency per affected `EthIfPhysController`

Status: DRAFT

⌈The Ethernet Interface Module shall check the MACsec related configuration on consistency per affected `EthIfPhysController`⌋

[SWS_EthIf_CONSTR_00013] Constraints for MACsec configuration

Status: DRAFT

⌈If an Ethernet hardware entity (e.g. PHY) has MACsec protection enabled, then EthIf shall check the corresponding configuration, such that MACsec in hardware is exclusively configured under either the following conditions (XOR):

- MACsec support is enabled on the `EthCtrlConfig` that is referenced by the affected `EthIfPhysController` via `EthIfEthCtrlRef`
- MACsec support is enabled on the `EthTrcvConfig` that is referenced by an `EthIfTransceiver` which in turn is referenced by an `EthIfController` that references the affected `EthIfPhysController` via `EthIfEthCtrlRef`
- MACsec support is enabled on the `EthSwtPort`s that belong to an `EthSwtConfig` which is referenced by an `EthIfSwitch` which in turn is referenced by an `EthIfController` that references the affected `EthIfPhysController` via `EthIfEthCtrlRef`
- MACsec support is enabled on the `EthSwtPort`s that belong to an `EthSwtConfig` which is referenced by an `EthIfSwitchPortGroup` which in turn is referenced by an `EthIfController` that references the affected `EthIfPhysController` via `EthIfEthCtrlRef`
- MACsec support is enabled in a `EthTrcvConfig` which referenced by an `EthSwtPort` that belongs to an `EthSwtConfig` which is referenced by an `EthIfSwitch` which in turn is referenced by an `EthIfController` that references the affected `EthIfPhysController` via `EthIfEthCtrlRef`
- MACsec support is enabled in a `EthTrcvConfig` which referenced by an `EthSwtPort` that belongs to an `EthSwtConfig` which is referenced by an `EthIfSwitchPortGroup` which in turn is referenced by an `EthIfController` that references the affected `EthIfPhysController` via `EthIfEthCtrlRef`

All other configuration constellations shall be rejected as invalid.⌋

[SWS_EthIf_CONSTR_00014] Constraint for MACsec configuration of EthIfPhysController

Status: DRAFT

⌈For usage of MACsec and with respect to [SWS_EthIf_CONSTR_00013], a configuration is only valid if the EthIfPhysController references an EthCtrlConfig. In all other cases the configuration shall be rejected.⌋

MACsec support is configured per MACsec protected Ethernet hardware entity (e.g. Ethernet controller, PHY or Ethernet switch port). The configuration for such an Ethernet hardware entity need to bypass MACsec protection for at least Ethernet frames used for the MKA protocol to establish a secured connection. This is ensured by bypassing Ethernet frames with EtherType set to the EAPOL Ethernet Type specified by [23, IEEE-802.1X-2020]. The bypassing rule is configured in the lower layer driver (e.g. Eth, EthTrcv or EthSwt) of the corresponding Ethernet hardware entity. Additionally, a EthIfPhysController that is associated with a MACsec protected Ethernet hardware entity need to have one EthIfController that process Ethernet frames with EtherType set to EAPOL Ethernet Type specified by [23, IEEE-802.1X-2020].

[SWS_EthIf_CONSTR_00015] Constraint for MACsec configuration regarding a EthIfController to process Ethernet frames with EtherType set to the EAPOL Ethernet Type specified by IEEE-802.1X

Status: DRAFT

⌈An EthIfPhysController that is associated with a MACsec protected Ethernet hardware entity (e.g. Ethernet controller or PHY) shall have one EthIfController where the following constraints are fulfilled:

- an EthIfFrameRxPool and EthIfFrameTxPool where both belongs to the same EthIfFrameConfig reference this EthIfController
- this EthIfFrameConfig has EthIfFrameType set to the value of the EAPOL Ethernet Type specified by IEEE-802.1X

⌋

Ethernet communication by applications shall only take place once MACsec secured communication is established. In the case of too early communication, messages might be discarded by the peer. Hence, EthIf indicates the affected upper layer (e.g. EthSM, BswM or configured users) regarding ETHTRCV_LINK_STATE_ACTIVE only when MACsec is operational (e.g. secured association is established) in the respective Ethernet hardware entity (e.g. PHY). The MACsec operational state is reported via EthIf_MacSecOperational per MkaPaeInstance. The state is given with API parameter MacSecOperational and either set to TRUE (the MACsec protected Ethernet hardware entity is operational) or set to FALSE (the MACsec protected Ethernet hardware entity is not operational). The MACsec operational state is considered for link state accumulation of EthIfSwitchPortGroup. Please refer to chapter Chap-

ter 7.1.10.3 “Link state handling” for more details on dependency between MACsec operational and reporting of link state changes.

[SWS_EthIf_00688] Indication of `EthIf_MacSecOperational` with a `CtrlIdx` that refers to an `EthIfController` that is referenced by a `MkaPaeInstance`

Status: DRAFT
Upstream requirements: FO_RS_MACsec_00002

⌈If `EthIf_MacSecOperational` is called with a `CtrlIdx` that addresses an `EthIfController` which is referenced by a `MkaPaeInstance`, then EthIf shall store the MACsec operational state for the corresponding Ethernet hardware entity (i.e. PHY) to share the state across all `EthIfController` that depend on this Ethernet hardware entity (i.e. PHY) and return with `E_OK`. Otherwise EthIf shall immediately return with `E_NOT_OK`.⌋

[SWS_EthIf_00689] Indication of `EthIf_SwitchMacSecOperational` with a `MgmtInfoPtr` where the given `EthSwtPort` is referenced by a `MkaPaeInstance`

Status: DRAFT
Upstream requirements: SRS_Eth_00033, FO_RS_MACsec_00002

⌈If `EthIf_SwitchMacSecOperational` is called and the following conditions are fulfilled:

- the given `MgmtInfoPtr` containes an `SwitchIdx` which idenfies an `EthIfSwitch` that is referenced by an `EthIfController`, or the given `MgmtInfoPtr` contains an `SwitchPortIdx` which is referenced by an configured `EthIfSwitchPortGroup` and this `EthIfSwitchPortGroup` is referenced by an `EthIfController`
- the given `EthSwtPort` is referenced by a `MkaPaeInstance`

then EthIf shall store the MACsec operational state for the given `EthSwtPort` and return with `E_OK`. Otherwise EthIf shall immediately return with `E_NOT_OK`.⌋

A MACsec protected Ethernet hardware entity is reflected in the configuration with a reference from a `MkaPaeInstance` to a specfic ECUC configuration element. Thereby some constraints need to be considered.

[SWS_EthIf_CONSTR_00016] PHY as MACsec protected Ethernet hardware entity

Status: DRAFT

⌈If the MACsec protected Ethernet hardware entity is a PHY, then an `EthIfController` shall be referenced by a `MkaPaeInstance`, where this `EthIfController` references an `EthIfTransceiver` and the corresponding `EthTrcvConfig` has `EthTrcvMacSecEnabled` set to `TRUE`.⌋

[SWS_EthIf_CONSTR_00017] Ethernet controller as MACsec protected Ethernet hardware entity

Status: DRAFT

⌈If the MACsec protected hardware entity is an Ethernet controller, then a `EthIfController` shall be referenced by a `MkaPaeInstance`, where this `EthIfController` references an `EthIfPhysController` where the corresponding `EthCtrlConfig` has `EthCtrlMacSecEnabled` set to `TRUE`.⌋

[SWS_EthIf_CONSTR_00018] Ethernet switch port as MACsec protected Ethernet hardware entity

Status: DRAFT

⌈If the MACsec protected hardware entity is an Ethernet switch port, then a `EthSwtPort` shall be referenced by a `MkaPaeInstance`, where `EthSwtPortMacSecEnabled` of this `EthSwtPort` is set to `TRUE`.⌋

If a link state change reporting towards the EthSM or configured user should consider the MACsec operational state of the corresponding MACsec protected Ethernet hardware (e.g. PHY or Ethernet switch port), then the affected `EthIfController` has to reference the `MkaPaeInstance` that belongs to this MACsec protected Ethernet hardware entity. Thereby some constraints need to be considered.

[SWS_EthIf_CONSTR_00019] `EthIfController` which reference an `EthIfTransceiver` shall reference at most one `MkaPaeInstance`

Status: DRAFT

⌈`EthIfController` which reference an `EthIfTransceiver` shall reference at most one `MkaPaeInstance`.⌋

[SWS_EthIf_CONSTR_00020] `EthIfController` which reference an `EthIfTransceiver` shall reference the same `MkaPaeInstance`

Status: DRAFT

⌈All `EthIfController` shall reference the same `MkaPaeInstance` which reference to the same `EthIfTransceiver` where the corresponding Ethernet hardware entity (PHY) is MACsec protected and the reporting towards EthSM and configured users need to consider the MACsec operational state of this Ethernet hardware entity.⌋

[SWS_EthIf_CONSTR_00021] `EthIfController` which reference an `EthIfPhysController` shall reference the same `MkaPaeInstance`

Status: DRAFT

⌈All `EthIfController` shall reference the same `MkaPaeInstance` which reference the same `EthIfPhysController` where the corresponding Ethernet hardware entity (Ethernet controller) is MACsec protected and the reporting towards EthSM and configured users need to consider the MACsec operational state of this Ethernet hardware entity.⌋

EthIf handles MACsec related API calls with a direct call towards to affected destination software driver (e.g. Eth driver) of the corresponding Ethernet hardware that hosts the MACsec protecting functionality. For a configured `EthTrcvConfig` (representing a PHY) that is referenced by an `EthSwtPort`, EthIf needs to scan the configuration and forward the APIs directly to the EthTrcv by considering the `TrcvIdx` in context of the EthTrcv driver.

[SWS_EthIf_00690] Forwarding of MACsec related API calls towards the Eth driver

Status: DRAFT
Upstream requirements: FO_RS_MACsec_00002

⌈If MACsec support is enabled on the `EthCtrlConfig` that is referenced by the affected `EthIfPhysController` via `EthIfEthCtrlRef` (see [SWS_EthIf_CONSTR_00013]) and one of the following MACsec APIs is called for an `EthIfController` that references this `EthIfPhysController` and is referenced by a `MkaPaeInstance`, then EthIf shall forward the API call to the Eth driver by considering the `CtrlIdx` in context of the Eth driver:

- a call of `EthIf_MacSecUpdateSecY` shall be forwarded to `Eth_MacSecUpdateSecY`
- a call of `EthIf_MacSecInitRxSc` shall be forwarded to `Eth_MacSecInitRxSc`
- a call of `EthIf_MacSecResetRxSc` shall be forwarded to `Eth_MacSecResetRxSc`
- a call of `EthIf_MacSecAddTxSa` shall be forwarded to `Eth_MacSecAddTxSa`
- a call of `EthIf_MacSecUpdateTxSa` shall be forwarded to `Eth_MacSecUpdateTxSa`
- a call of `EthIf_MacSecDeleteTxSa` shall be forwarded to `Eth_MacSecDeleteTxSa`
- a call of `EthIf_MacSecAddRxSa` shall be forwarded to `Eth_MacSecAddRxSa`
- a call of `EthIf_MacSecUpdateRxSa` shall be forwarded to `Eth_MacSecUpdateRxSa`
- a call of `EthIf_MacSecDeleteRxSa` shall be forwarded to `Eth_MacSecDeleteRxSa`
- a call of `EthIf_MacSecGetTxSaNextPn` shall be forwarded to `Eth_MacSecGetTxSaNextPn`
- a call of `EthIf_MacSecSetControlledPortEnabled` shall be forwarded to `Eth_MacSecSetControlledPortEnabled`
- a call of `EthIf_MacSecGetMacSecStatistics` shall be forwarded to `Eth_MacSecGetMacSecStatistics`

Otherwise an API call shall return with E_NOT_OK.⌋

[SWS_EthIf_00691] Forwarding of MACsec related API calls towards the EthTrcv driver

Status: DRAFT
Upstream requirements: FO_RS_MACsec_00002

⌈If MACsec support is enabled on the EthTrcvConfig that is referenced by an EthIfTransceiver which in turn is referenced by an EthIfController that references the affected EthIfPhysController via EthIfEthCtrlRef (see [SWS_EthIf_CONSTR_00013]) and one of the following MACsec APIs is called for the EthIfController that references this EthIfTransceiver and is referenced by a MkaPaeInstance, then EthIf shall forward the API call to the EthTrcv driver by considering the TrcvIdx in context of the EthTrcv driver:

- a call of EthIf_MacSecUpdateSecY shall be forwarded to EthTrcv_MacSecUpdateSecY
- a call of EthIf_MacSecInitRxSc shall be forwarded to EthTrcv_MacSecInitRxSc
- a call of EthIf_MacSecResetRxSc shall be forwarded to EthTrcv_MacSecResetRxSc
- a call of EthIf_MacSecAddTxSa shall be forwarded to EthTrcv_MacSecAddTxSa
- a call of EthIf_MacSecUpdateTxSa shall be forwarded to EthTrcv_MacSecUpdateTxSa
- a call of EthIf_MacSecDeleteTxSa shall be forwarded to EthTrcv_MacSecDeleteTxSa
- a call of EthIf_MacSecAddRxSa shall be forwarded to EthTrcv_MacSecAddRxSa
- a call of EthIf_MacSecUpdateRxSa shall be forwarded to EthTrcv_MacSecUpdateRxSa
- a call of EthIf_MacSecDeleteRxSa shall be forwarded to EthTrcv_MacSecDeleteRxSa
- a call of EthIf_MacSecGetTxSaNextPn shall be forwarded to EthTrcv_MacSecGetTxSaNextPn
- a call of EthIf_MacSecSetControlledPortEnabled shall be forwarded to EthTrcv_MacSecSetControlledPortEnabled
- a call of EthIf_MacSecGetMacSecStatistics shall be forwarded to EthTrcv_MacSecGetMacSecStatistics

Otherwise a API call shall return with E_NOT_OK⌋

[SWS_EthIf_00692] Forwarding of MACsec related API calls towards the EthTrcv driver where a `EthTrcvConfig` is referenced by an `EthSwtPort`

Status: DRAFT
Upstream requirements: FO_RS_MACsec_00002

⌈If MACsec support is enabled via either the following configurations (see [SWS_EthIf_CONSTR_00013]):

- MACsec support is enabled on the `EthTrcvConfig` which is referenced by an `EthSwtPort` that belongs to an `EthSwtConfig` which is referenced by an `EthIfSwitch` which in turn is referenced by an `EthIfController` that references the affected `EthIfPhysController` via `EthIfEthCtrlRef`
- MACsec support is enabled on the `EthTrcvConfig` which referenced by an `EthSwtPort` that belong to an `EthSwtConfig` which is referenced by an `EthIfSwitchPortGroup` which in turn is referenced by an `EthIfController` that references the affected `EthIfPhysController` via `EthIfEthCtrlRef`
- The `EthSwtPort` is referenced by an `EthIfSwitchPortGroup` which in turn is referenced by an `EthIfController` that references the affected `EthIfPhysController` via `EthIfEthCtrlRef`
- the `EthSwtPort` belongs to an `EthIfSwitch` that is in turn referenced by an `EthIfController` that references the affected `EthIfPhysController` via `EthIfEthCtrlRef`

and one of the following MACsec APIs is called for an `EthSwtPort` that is referenced by the `EthIfSwitch` or the `EthIfSwitchPortGroup` and this `EthSwtPort` is referenced by a `MkaPaeInstance`, then EthIf shall forward the API call to the EthTrcv driver by considering the `TrcvIdx` in context of the EthTrcv driver:

- a call of `EthIf_SwitchMacSecUpdateSecY` shall be forwarded to `EthSwt_MacSecUpdateSecY`
- a call of `EthIf_SwitchMacSecUpdateSecY` shall be forwarded to `EthTrcv_MacSecUpdateSecY`
- a call of `EthIf_SwitchMacSecInitRxSc` shall be forwarded to `EthTrcv_MacSecInitRxSc`
- a call of `EthIf_SwitchMacSecResetRxSc` shall be forwarded to `EthTrcv_MacSecResetRxSc`
- a call of `EthIf_SwitchMacSecAddTxSa` shall be forwarded to `EthTrcv_MacSecAddTxSa`
- a call of `EthIf_SwitchMacSecUpdateTxSa` shall be forwarded to `EthTrcv_MacSecUpdateTxSa`
- a call of `EthIf_SwitchMacSecDeleteTxSa` shall be forwarded to `EthTrcv_MacSecDeleteTxSa`

- a call of EthIf_SwitchMacSecAddRxSa shall be forwarded to EthTrcv_MacSecAddRxSa
- a call of EthIf_SwitchMacSecUpdateRxSa shall be forwarded to EthTrcv_MacSecUpdateRxSa
- a call of EthIf_SwitchMacSecDeleteRxSa shall be forwarded to EthTrcv_MacSecDeleteRxSa
- a call of EthIf_SwitchMacSecGetTxSaNextPn shall be forwarded to EthTrcv_MacSecGetTxSaNextPn
- a call of EthIf_SwitchMacSecSetControlledPortEnabled shall be forwarded to EthTrcv_MacSecSetControlledPortEnabled
- a call of EthIf_SwitchMacSecGetMacSecStatistics shall be forwarded to EthTrcv_MacSecGetMacSecStatistics

Otherwise a API call shall return with E_NOT_OK.⌋

[SWS_EthIf_00693] Forwarding of MACsec related API calls towards the EthSwt driver where the EthSwtPort has MACsec support enabled

Status: DRAFT
Upstream requirements: FO_RS_MACsec_00002

⌈If MACsec support is enabled on the EthSwtPort and either one of the following configurations is valid (see [SWS_EthIf_CONSTR_00013]):

- The EthSwtPort is referenced by an EthIfSwitchPortGroup which in turn is referenced by an EthIfController that references the affected EthIfPhysController via EthIfEthCtrlRef
- the EthSwtPort belongs to an EthIfSwitch that is in turn referenced by an EthIfController that references the affected EthIfPhysController via EthIfEthCtrlRef

and one of the following MACsec APIs is called for an EthSwtPort that is referenced by the EthIfSwitch or the EthIfSwitchPortGroup and this EthSwtPort is referenced by a MkaPaeInstance, then EthIf shall forward the API call to the EthSwt driver by considering the SwitchIdx in context of the EthSwt driver:

- a call of EthIf_SwitchMacSecUpdateSecY shall be forwarded to EthSwt_MacSecUpdateSecY
- a call of EthIf_SwitchMacSecInitRxSc shall be forwarded to EthSwt_MacSecInitRxSc
- a call of EthIf_SwitchMacSecResetRxSc shall be forwarded to EthSwt_MacSecResetRxSc
- a call of EthIf_SwitchMacSecAddTxSa shall be forwarded to EthSwt_MacSecAddTxSa

- a call of EthIf_SwitchMacSecUpdateTxSa shall be forwarded to EthSwt_MacSecUpdateTxSa
- a call of EthIf_SwitchMacSecDeleteTxSa shall be forwarded to EthSwt_MacSecDeleteTxSa
- a call of EthIf_SwitchMacSecAddRxSa shall be forwarded to EthSwt_MacSecAddRxSa
- a call of EthIf_SwitchMacSecUpdateRxSa shall be forwarded to EthSwt_MacSecUpdateRxSa
- a call of EthIf_SwitchMacSecDeleteRxSa shall be forwarded to EthSwt_MacSecDeleteRxSa
- a call of EthIf_SwitchMacSecGetTxSaNextPn shall be forwarded to EthSwt_MacSecGetTxSaNextPn
- a call of EthIf_SwitchMacSecSetControlledPortEnabled shall be forwarded to EthSwt_MacSecSetControlledPortEnabled
- a call of EthIf_SwitchMacSecGetMacSecStatistics shall be forwarded to EthSwt_MacSecGetMacSecStatistics

Otherwise a API call shall return with E_NOT_OK⌋

[SWS_EthIf_00694] Forwarding of MACsec related callback APIs calls towards the MKA module called by the Eth driver

Status: DRAFT
Upstream requirements: FO_RS_MACsec_00002

⌈If MACsec support is enabled on the EthCtrlConfig that is referenced by the affected EthIfPhysController via EthIfEthCtrlRef (see [SWS_EthIf_CONSTR_00013]) and one of the following MACsec callback APIs is called for this EthIfPhysController, then EthIf shall forward this call towards the MKA module by considering the EthIfController that referencing this EthIfPhysController and the MkaPaeIdx which references this EthIfController:

- a call of EthIf_EthMacSecUpdateSecYNotification shall be forwarded to Mka_MacSecUpdateSecYNotification
- a call of EthIf_EthMacSecAddTxSaNotification shall be forwarded to Mka_MacSecAddTxSaNotification
- a call of EthIf_EthMacSecAddRxSaNotification shall be forwarded to Mka_MacSecAddRxSaNotification
- a call of EthIf_EthMacSecGetMacSecStatisticsNotification shall be forwarded to Mka_GetMacSecStatisticsNotification

Otherwise a callback API call shall be ignored.⌋

[SWS_EthIf_00695] Forwarding of MACsec related callback APIs calls towards the MKA module called by the EthTrcv driver

Status: DRAFT
Upstream requirements: FO_RS_MACsec_00002

⌈If MACsec support is enabled on the `EthTrcvConfig` that is referenced by an `EthIfTransceiver` which in turn is referenced by an `EthIfController` that references the affected `EthIfPhysController` via `EthIfEthCtrlRef` (see [SWS_EthIf_CONSTR_00013]) and one of the following MACsec callback APIs is called for an `EthIfController` that references this `EthIfTransceiver` and this `EthIfController` is referenced by a `MkaPaeInstance`, then EthIf shall forward the API call towards the MKA module by considering the `MkaPaeIdx` in context of the MKA module. Otherwise a callback API call shall be ignored:

- a call of `EthIf_MacSecUpdateSecYNotification` shall be forwarded to `Mka_MacSecUpdateSecYNotification`
- a call of `EthIf_MacSecAddTxSaNotification` shall be forwarded to `Mka_MacSecAddTxSaNotification`
- a call of `EthIf_MacSecAddRxSaNotification` shall be forwarded to `Mka_MacSecAddRxSaNotification`
- a call of `EthIf_MacSecGetMacSecStatisticsNotification` shall be forwarded to `Mka_GetMacSecStatisticsNotification`

⌋

[SWS_EthIf_00696] Forwarding of MACsec related callback APIs calls towards the MKA module called by the EthTrcv driver where the `EthTrcvConfig` is referenced by an `EthSwtPort`

Status: DRAFT
Upstream requirements: FO_RS_MACsec_00002

⌈If MACsec support is enabled via either the following configurations (see [SWS_EthIf_CONSTR_00013]):

- MACsec support is enabled on the `EthTrcvConfig` which is referenced by an `EthSwtPort` that belong to an `EthSwtConfig` which is referenced by an `EthIfSwitch` which in turn is referenced by an `EthIfController` that references the affected `EthIfPhysController` via `EthIfEthCtrlRef`
- MACsec support is enabled on the `EthTrcvConfig` which referenced by an `EthSwtPort` that belong to an `EthSwtConfig` which is referenced by an `EthIfSwitchPortGroup` which in turn is referenced by an `EthIfController` that references the affected `EthIfPhysController` via `EthIfEthCtrlRef`

and one of the following MACsec callback APIs is called for an `EthSwtPort` that references the `EthIfSwitch` or the `EthIfSwitchPortGroup` and this `EthSwtPort` is referenced by a `MkaPaeInstance`, then EthIf shall forward the callback API call

towards the MKA module by considering the `MkaPaeIdx` in context of the MKA module. Otherwise a callback API call shall be ignored:

- a call of `EthIf_SwitchMacSecUpdateSecYNotification` shall be forwarded to `Mka_MacSecUpdateSecYNotification`
- a call of `EthIf_SwitchMacSecAddTxSaNotification` shall be forwarded to `Mka_MacSecAddTxSaNotification`
- a call of `EthIf_SwitchMacSecAddRxSaNotification` shall be forwarded to `Mka_MacSecAddRxSaNotification`
- a call of `EthIf_SwitchMacSecGetMacSecStatisticsNotification` shall be forwarded to `Mka_GetMacSecStatisticsNotification`

⌋

[SWS_EthIf_00583]
Status: DRAFT
Upstream requirements: FO_RS_MACsec_00023

⌈The Ethernet Interface module shall support the MKA related EtherTypes as defined in [24].⌋

[SWS_EthIf_00584]
Status: DRAFT

⌈The Ethernet Interface module shall allow forwarding the received Ethernet frames of a specific EtherType to multiple frame owners if configured.⌋

7.1.16.1 MACsec support in software

[SWS_EthIf_00560]
Status: DRAFT
Upstream requirements: FO_RS_MACsec_00001

⌈The Ethernet Interface shall support MACsec as a SW implementation as specified in [24].⌋

[SWS_EthIf_00561]
Status: DRAFT
Upstream requirements: FO_RS_MACsec_00004

⌈The Ethernet Interface shall support configuring which Ethernet Interface Controllers are MACsec protected.⌋

[SWS_EthIf_00562]
Status: DRAFT
Upstream requirements: FO_RS_MACsec_00004

⌈The Ethernet Interface shall support configuring per Ethernet Interface Controller the MACsec Entity to use (per SW or HW i.e., offloaded).⌋

Note: This is included per configuration with the parameter `EthIfMacSecSupport`.

[SWS_EthIf_00563]
Status: DRAFT
Upstream requirements: FO_RS_MACsec_00007

⌈The MACsec Entity per SW of the Ethernet Interface shall provide a mechanism to configure rules to bypass MACsec for incoming and outgoing traffic based on EtherType and/or VLAN-ID. All traffic not configured as bypassed traffic shall be processed by the MACsec entity or dropped. This configuration shall be supported at initial configuration time of the Ports.⌋

[SWS_EthIf_00564]
Status: DRAFT
Upstream requirements: FO_RS_MACsec_00009

⌈The MACsec entity per SW of the Ethernet Interface shall support status counters for the following information, which may be attached to IDSM functionality:

- Dropped frames because of incorrect ICV per port.
- Unsuccessful MKA sequence per peer.
- Additionally, all the port statistics required by [24].

⌋

[SWS_EthIf_00565]
Status: DRAFT
Upstream requirements: FO_RS_MACsec_00010

⌈The MACsec entity per SW of the Ethernet Interface shall support "Integrity only" as well as "Integrity with Confidentiality" for all supported ciphers.⌋

[SWS_EthIf_00566]
Status: DRAFT
Upstream requirements: FO_RS_MACsec_00011

⌈The MACsec entity per SW of the Ethernet Interface shall support MAC Security TAG (SecTAG) as defined in [24]. The SecTAG shall convey:

- TAG Control Information (TCI)
- Association Number (AN)
- Short Length (SL)

- Packet Number (PN)
- Secure Channel Identifier (SCI) - Optional

⌋

[SWS_EthIf_00567]
Status: DRAFT
Upstream requirements: FO_RS_MACsec_00012

⌈The MACsec entity per SW of the Ethernet Interface shall support MACsec EtherType as defined in [24].⌋

[SWS_EthIf_00568]
Status: DRAFT
Upstream requirements: FO_RS_MACsec_00011

⌈The MACsec entity per SW of the Ethernet Interface shall support TAG Control Information (TCI) as defined in [24]. The TCI shall be encoded in the SecTAG.⌋

[SWS_EthIf_00569]
Status: DRAFT
Upstream requirements: FO_RS_MACsec_00011

⌈The MACsec entity per SW of the Ethernet Interface shall support Association Number (AN) as defined in [24]. The AN shall be encoded in the SecTAG.⌋

[SWS_EthIf_00570]
Status: DRAFT
Upstream requirements: FO_RS_MACsec_00011

⌈The MACsec entity per SW of the Ethernet Interface shall support Short Length (SL) as defined in [24]. The SL shall be encoded in the SecTAG.⌋

[SWS_EthIf_00571]
Status: DRAFT
Upstream requirements: FO_RS_MACsec_00011

⌈The MACsec entity per SW of the Ethernet Interface shall support Packet Number (PN) with 32 least significant bits, as defined in [24]. The PN shall be encoded in the SecTAG.⌋

[SWS_EthIf_00572]
Status: DRAFT
Upstream requirements: FO_RS_MACsec_00017

⌈The MACsec entity per SW of the Ethernet Interface shall support Extended Packet Number (XPN) as defined in [24]. The XPN extends the PN to 64 bits.⌋

[SWS_EthIf_00573]
Status: DRAFT
Upstream requirements: FO_RS_MACsec_00018

⌈The MACsec entity per SW of the Ethernet Interface shall support Secure Channel Identifier (SCI), as defined in [24]. The SCI may be encoded in the SecTAG if SCI is required to be sent.⌋

[SWS_EthIf_00574]
Status: DRAFT
Upstream requirements: FO_RS_MACsec_00019

⌈The MACsec entity per SW of the Ethernet Interface shall support Secure Data as defined in [24].⌋

[SWS_EthIf_00575]
Status: DRAFT
Upstream requirements: FO_RS_MACsec_00020

⌈The MACsec entity per SW of the Ethernet Interface shall support Integrity check value (ICV) as defined in [24]. The ICV length depends on the used cipher suite but is not less than 8 octets and not more than 16 octets. The transmitted ICV is always 16 octets.⌋

[SWS_EthIf_00576]
Status: DRAFT
Upstream requirements: FO_RS_MACsec_00021

⌈The MACsec entity per SW of the Ethernet Interface shall support a protect function as specified in [24].⌋

[SWS_EthIf_00577]
Status: DRAFT
Upstream requirements: FO_RS_MACsec_00022

⌈The MACsec entity per SW of the Ethernet Interface shall support a validation function as specified in [24].⌋

[SWS_EthIf_00578]
Status: DRAFT
Upstream requirements: FO_RS_MACsec_00032

⌈The MACsec entity per SW of the Ethernet Interface shall support the following ciphers suites:

- GCM-AES-128
- GCM-AES-256
- GCM-AES-XPN-128
- GCM-AES-XPN-256

⌋

[SWS_EthIf_00579]
Status: DRAFT
Upstream requirements: FO_RS_MACsec_00033

⌈The MACsec entity per SW of the Ethernet Interface shall support a validation function for MACsec ICV.⌋

[SWS_EthIf_00580]
Status: DRAFT
Upstream requirements: FO_RS_MACsec_00034

⌈The MACsec entity per SW of the Ethernet Interface shall support a generation function for MACsec ICV.⌋

7.1.17 Firewall support

The Ethernet stack supports firewalling of network packets by means of the firewall module. The firewall support is managed by the parameter `EthIfFwSupport`.

[SWS_EthIf_00668] Handling if `EthIfFwSupport` is set to `FIREWALL_WITHOUT_PERSTREAMFILTERING`

Status: DRAFT

⌈If `EthIfFwSupport` is set to `FIREWALL_WITHOUT_PERSTREAMFILTERING`, EthIf shall set `FILTER_RULE_ID_16` in the PDU MetaData to 0xFFFF.⌋

[SWS_EthIf_00669] Handling if `EthIfFwSupport` is set to `FIREWALL_WITH_PERSTREAMFILTERING`

Status: DRAFT

⌈If `EthIfFwSupport` is set to `FIREWALL_WITH_PERSTREAMFILTERING`, EthIf shall call `EthSwt_ExtractStreamHandleIdx` to set the `FILTER_RULE_ID_16` in the PDU MetaData to the value stored at the `StreamHandleIdxPtr`.⌋

Call forwarding to the switch

The firewall has some interactions with the Ethernet Switch Driver for some use-cases. The EthIf module has thus to forward these calls.

[SWS_EthIf_00592]

Status: DRAFT

⌈If `EthIf_GetStreamStatistics` is called, the EthIf module shall call `EthSwt_GetStreamStatistics` with the same parameters.⌋

[SWS_EthIf_00662] Forwarding of stream statistics indications to firewall module

Status: DRAFT

⌈When the respective callback function `EthIf_StreamStatisticsIndication` is called, the EthIf module shall call `Fw_StreamStatisticsIndication` with the same parameters.⌋

[SWS_EthIf_00593]

Status: DRAFT

⌈If `EthIf_SetStreamState` is called, the EthIf module shall call `EthSwt_SetStreamState` with the same parameters. When the respective callback function `EthIf_StreamStateIndication` is called, the EthIf module shall call `Fw_StreamStateIndication` with the same parameters.⌋

7.2 Security Events

[SWS_EthIf_00502]

Status: DRAFT
Upstream requirements: RS_Ids_00810

⌈If security event reporting has been enabled for the EthIf module (EthIfEnableSecurityEventReporting = true) the respective security events shall be reported to the IdsM via the interfaces defined in AUTOSAR_SWS_BSWGeneral [6].⌋

The following table lists the security events which are standardized for the EthIf together with their trigger conditions:

[SWS_EthIf_00503] Security events for EthIf

Status: DRAFT
Upstream requirements: RS_Ids_00810

⌈

Name	*Description*	*ID*
SEV_ETH_DROP_UNKNOWN_ETHERTYPE	An ethernet datagram was dropped due the Ethertype is not known.	15
SEV_ETH_DROP_VLAN_DOUBLE_TAG	An ethernet datagram was dropped due to double VLAN tag.	16
SEV_ETH_DROP_INV_VLAN	An ethernet datagram was dropped due to an invalid Crtl Idx/VLAN.	17
SEV_ETH_DROP_MAC_COLLISION	Ethernet datagram was dropped because local MAC was same as source MAC in an incoming frame.	18

⌋

[SWS_EthIf_00698] Trigger condition of SEV_ETH_DROP_UNKNOWN_ETHERTYPE

Status: DRAFT
Upstream requirements: RS_Ids_00810

⌈Upon dropping of an ethernet datagram due to the EtherType is not known, EthIf shall raise SEV_ETH_DROP_UNKNOWN_ETHERTYPE to the IdsM.⌋

[SWS_EthIf_00699] Security event context data definition: SEV_ETH_DROP_UNKNOWN_ETHERTYPE

Status: DRAFT
Upstream requirements: RS_Ids_00810

⌈

SEV Name	***SEV_ETH_DROP_UNKNOWN_ETHERTYPE***	
ID	15	
Description	An ethernet datagram was dropped due the Ethertype is not known.	
Context Data Version	1	
Context Data	***Data Type***	***Allowed Values***
Length	uint16	Length of EthernetFrame byte array
EthernetFrame	uint8 [54]	EthernetFrame, truncated to the first EthIfSEvEthernet FrameMaxLength bytes
	MAX-LENGTH	Truncation length can be defined on a project specific basis. AUTOSAR has defined a default value, as given in the EthernetFrame byte array definition above.

⌋

[SWS_EthIf_00700] Trigger condition of SEV_ETH_DROP_VLAN_DOUBLE_TAG

Status: DRAFT
Upstream requirements: RS_Ids_00810

⌈Upon dropping of an ethernet datagram due to double VLAN tag, EthIf shall raise SEV_ETH_DROP_VLAN_DOUBLE_TAG to the IdsM.⌋

[SWS_EthIf_00701] Security event context data definition: SEV_ETH_DROP_VLAN_DOUBLE_TAG

Status: DRAFT
Upstream requirements: RS_Ids_00810

⌈

SEV Name	***SEV_ETH_DROP_VLAN_DOUBLE_TAG***	
ID	16	
Description	An ethernet datagram was dropped due to double VLAN tag.	
Context Data Version	1	
Context Data	***Data Type***	***Allowed Values***
Length	uint16	Length of EthernetFrame byte array
EthernetFrame	uint8 [54]	EthernetFrame, truncated to the first EthIfSEvEthernet FrameMaxLength bytes

▽

△

SEV Name	*SEV_ETH_DROP_VLAN_DOUBLE_TAG*	
	MAX-LENGTH	Truncation length can be defined on a project specific basis. AUTOSAR has defined a default value, as given in the EthernetFrame byte array definition above.

⌋

[SWS_EthIf_00702] Trigger condition of SEV_ETH_DROP_INV_VLAN

Status: DRAFT
Upstream requirements: RS_Ids_00810

⌈Upon dropping of an ethernet datagram due to an invalid CrtlIdx/VLAN, EthIf shall raise SEV_ETH_DROP_INV_VLAN to the IdsM.⌋

[SWS_EthIf_00703] Security event context data definition: SEV_ETH_DROP_INV_VLAN

Status: DRAFT
Upstream requirements: RS_Ids_00810

⌈

SEV Name	*SEV_ETH_DROP_INV_VLAN*	
ID	17	
Description	An ethernet datagram was dropped due to an invalid CrtlIdx/VLAN.	
Context Data Version	1	
Context Data	*Data Type*	*Allowed Values*
Length	uint16	Length of EthernetFrame byte array
EthernetFrame	uint8 [54]	EthernetFrame, truncated to the first EthIfSEvEthernet FrameMaxLength bytes
	MAX-LENGTH	Truncation length can be defined on a project specific basis. AUTOSAR has defined a default value, as given in the EthernetFrame byte array definition above.

⌋

[SWS_EthIf_00704] Trigger condition of SEV_ETH_DROP_MAC_COLLISION

Status: DRAFT
Upstream requirements: RS_Ids_00810

⌈Upon dropping of an ethernet datagram due to the local MAC was same as the source MAC in an incoming frame, EthIf shall raise SEV_ETH_DROP_MAC_COLLISION to the IdsM.⌋

[SWS_EthIf_00705] Security event context data definition: SEV_ETH_DROP_MAC_COLLISION

Status: DRAFT
Upstream requirements: RS_Ids_00810

⌈

SEV Name	***SEV_ETH_DROP_MAC_COLLISION***	
ID	18	
Description	Ethernet datagram was dropped because local MAC was same as source MAC in an incoming frame.	
Context Data Version	1	
Context Data	***Data Type***	***Allowed Values***
Length	uint16	Length of EthernetFrame byte array
EthernetFrame	uint8 [54]	EthernetFrame, truncated to the first EthIfSEvEthernet FrameMaxLength bytes
	MAX-LENGTH	Truncation length can be defined on a project specific basis. AUTOSAR has defined a default value, as given in the EthernetFrame byte array definition above.

⌋

7.3 Error Classification

Chapter [6, General Specification of Basic Software Modules] 7.2 *"Error Handling"* describes the error handling of the Basic Software in detail. Above all, it constitutes a classification scheme consisting of five error types which may occur in BSW modules.

Based on this foundation, the following section specifies particular errors arranged in the respective subsections below.

7.3.1 Development Errors

[SWS_EthIf_00017] Definition of development errors in module EthIf ⌈

Type of error	***Related error code***	***Error value***
API service called with invalid controller index	ETHIF_E_INV_CTRL_IDX	0x01
API service called with invalid transceiver index	ETHIF_E_INV_TRCV_IDX	0x02
API service called with invalid switch index	ETHIF_E_INV_SWT_IDX	0x03
API service called with invalid port group index	ETHIF_E_INV_PORT_GROUP_IDX	0x04
API service called when EthIf module was not initialized	ETHIF_E_UNINIT	0x05
API service called with invalid pointer in parameter list	ETHIF_E_PARAM_POINTER	0x06
API service called with invalid parameter	ETHIF_E_INV_PARAM	0x07
EthIf_Init called with an invalid configuration pointer	ETHIF_E_INIT_FAILED	0x08
API service called with invalid switch port index	ETHIF_E_INV_PORT_IDX	0x09

▽

△

Type of error	*Related error code*	*Error value*
API service called with invalid clock unit index	ETHIF_E_INV_CLKUNIT_IDX	0x10

⌋

7.3.2 Runtime Errors

[SWS_EthIf_91136] Definition of runtime errors in module EthIf

Upstream requirements: SRS_BSW_00385

⌈

Type of error	*Related error code*	*Error value*
A PDU is requested to be used while it is already in use or requested to be available while it is already availalbe **Tags:** atp.Status=draft	ETHIF_E_PDU_STATE_TRANSITION_FAILED	0x01

⌋

7.3.3 Production Errors

There are no production errors.

7.3.4 Extended Production Errors

There are no extended production errors.

8 API specification

8.1 API Parameter Checking

[SWS_EthIf_00681] Handling unavailable forwarding APIs

Upstream requirements: SRS_BSW_00461

⌈When a forwarding API of EthIf is called that is not available in the responsible driver (e.g. Eth, EthTrcv, or EthSwt), the EthIf shall return immediately from the call. In case a return value is available and of type Std_ReturnType, it shall be set to `E_NOT_OK`.⌋

[SWS_EthIf_00652] DET error reporting of `ETHIF_E_PARAM_POINTER`

Upstream requirements: SRS_BSW_00323, SRS_BSW_00337

⌈The `EthIf` module reports the development error `ETHIF_E_PARAM_POINTER` when a NULL_PTR is not accepted as an argument to a service or callback function. The exact behavior is specified in [SWS_BSW_00050] and [SWS_BSW_00212]⌋

[SWS_EthIf_00653] DET error reporting of `ETHIF_E_INV_CTRL_IDX`

Upstream requirements: SRS_BSW_00323, SRS_BSW_00337

⌈The `EthIf` APIs which has the parameter `CtrlIdx` shall check the parameter `CtrlIdx` for being valid. If the check fails, the function shall raise the development error `ETHIF_E_INV_CTRL_IDX` when the development error detection is enabled.⌋

[SWS_EthIf_00654] DET error reporting of `ETHIF_E_INV_TRCV_IDX`

Upstream requirements: SRS_BSW_00323, SRS_BSW_00337

⌈The `EthIf` APIs which has the parameter `TrcvIdx` shall check the parameter `TrcvIdx` for being valid. If the check fails, the function shall raise the development error `ETHIF_E_INV_TRCV_IDX` when the development error detection is enabled.⌋

[SWS_EthIf_00655] DET error reporting of `ETHIF_E_INV_SWT_IDX`

Upstream requirements: SRS_BSW_00323, SRS_BSW_00337

⌈The `EthIf` APIs which has the parameter `SwitchIdx` shall check the parameter `SwitchIdx` for being valid. If the check fails, the function shall raise the development error `ETHIF_E_INV_SWT_IDX` when the development error detection is enabled.⌋

[SWS_EthIf_00656] DET error reporting of `ETHIF_E_INV_PORT_GROUP_IDX`

Upstream requirements: SRS_BSW_00323, SRS_BSW_00337

⌈The `EthIf` APIs which has the parameter `PortGroupIdx` shall check the parameter `PortGroupIdx` for being valid. If the check fails, the function shall raise the development error `ETHIF_E_INV_PORT_GROUP_IDX` when the development error detection is enabled.⌋

[SWS_EthIf_00657] DET error reporting of ETHIF_E_INV_PORT_IDX

Upstream requirements: SRS_BSW_00323, SRS_BSW_00337

⌈The EthIf APIs which has the parameter SwitchPortIdx shall check the parameter SwitchPortIdx for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_PORT_IDX when the development error detection is enabled.⌋

[SWS_EthIf_00658] DET error reporting of ETHIF_E_INV_PARAM

Upstream requirements: SRS_BSW_00323, SRS_BSW_00337

⌈The EthIf APIs which has the parameter BufIdx shall check the parameter BufIdx for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_PARAM when the development error detection is enabled.⌋

[SWS_EthIf_00679] DET error reporting of ETHIF_E_INV_CLKUNIT_IDX

Upstream requirements: SRS_BSW_00323, SRS_BSW_00337

⌈The EthIf APIs which has the parameter ClkUnitIdx shall check the parameter ClkUnitIdx for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_CLKUNIT_IDX when the development error detection is enabled.⌋

8.2 Imported types

This chapter lists all types included from the following module:

[SWS_EthIf_00023] Definition of imported datatypes of module EthIf ⌈

Module	*Header File*	*Imported Type*
Comtype	ComStack_Types.h	BufReq_ReturnType
	ComStack_Types.h	ListElemStructType (draft)
	ComStack_Types.h	PduIdType
	ComStack_Types.h	PduInfoType
	ComStack_Types.h	PduLengthType
	ComStack_Types.h	TimeStampQualType (draft)
	ComStack_Types.h	TimeStampType (draft)
	ComStack_Types.h	TimeTupleType (draft)
CV2x	CV2x_GeneralTypes.h	CV2x_BufCV2xPC5RxParamIdType (draft)
	CV2x_GeneralTypes.h	CV2x_BufCV2xPC5TxParamIdType (draft)
	CV2x_GeneralTypes.h	CV2x_GetChanTxParamIdType (draft)
EcuM	EcuM.h	EcuM_WakeupSourceType
Eth	Eth_GeneralTypes.h	Eth_BufIdxType
	Eth_GeneralTypes.h	Eth_CounterType
	Eth_GeneralTypes.h	Eth_DataType

▽

△

Module	*Header File*	*Imported Type*
	Eth_GeneralTypes.h	Eth_FilterActionType
	Eth_GeneralTypes.h	Eth_FrameType
	Eth_GeneralTypes.h	Eth_MacVlanType
	Eth_GeneralTypes.h	Eth_ModeType
	Eth_GeneralTypes.h	Eth_RxStatsType
	Eth_GeneralTypes.h	Eth_RxStatusType
	Eth_GeneralTypes.h	Eth_SpiStatusType (draft)
	Eth_GeneralTypes.h	Eth_StreamStatisticCounterType
	Eth_GeneralTypes.h	Eth_TxErrorCounterValuesType
	Eth_GeneralTypes.h	Eth_TxStatsType
EthSwt	Eth_GeneralTypes.h	EthSwt_MacLearningType
	Eth_GeneralTypes.h	EthSwt_MgmtInfoType
	Eth_GeneralTypes.h	EthSwt_MgmtObjectType
	Eth_GeneralTypes.h	EthSwt_MgmtObjectValidType
	Eth_GeneralTypes.h	EthSwt_MgmtOwner
	Eth_GeneralTypes.h	EthSwt_PortMirrorCfgType
	Eth_GeneralTypes.h	EthSwt_PortMirrorStateType
EthTrcv	Eth_GeneralTypes.h	EthTrcv_BaudRateType
	Eth_GeneralTypes.h	EthTrcv_CableDiagResultType
	Eth_GeneralTypes.h	EthTrcv_DuplexModeType
	Eth_GeneralTypes.h	EthTrcv_LinkStateType
	Eth_GeneralTypes.h	EthTrcv_MacMethodType (draft)
	Eth_GeneralTypes.h	EthTrcv_PhyLoopbackModeType
	Eth_GeneralTypes.h	EthTrcv_PhyTestModeType
	Eth_GeneralTypes.h	EthTrcv_PhyTxModeType
	Eth_GeneralTypes.h	EthTrcv_WakeupReasonType
Mka	Mka.h	Mka_ConfidentialityOffsetType (draft)
	Mka.h	Mka_MacSecConfigType (draft)
	Mka.h	Mka_SakKeyPtrType (draft)
	Mka.h	Mka_Stats_Rx_ScType (draft)
	Mka.h	Mka_Stats_Rx_SecYType (draft)
	Mka.h	Mka_Stats_SecYType (draft)
	Mka.h	Mka_Stats_Tx_ScType (draft)
	Mka.h	Mka_Stats_Tx_SecYType (draft)
	Mka.h	Mka_ValidateFramesType (draft)
Std	Std_Types.h	Std_ReturnType
	Std_Types.h	Std_VersionInfoType
WEth	WEth_GeneralTypes.h	WEth_BufWRxParamIdType
	WEth_GeneralTypes.h	WEth_BufWTxParamIdType
WEthTrcv	WEth_GeneralTypes.h	WEthTrcv_BandwidthType

▽

△

Module	Header File	Imported Type
	WEth_GeneralTypes.h	WEthTrcv_GetChanRxParamIdType
	WEth_GeneralTypes.h	WEthTrcv_RadioModeType
	WEth_GeneralTypes.h	WEthTrcv_RssiType
	WEth_GeneralTypes.h	WEthTrcv_SetChanRxParamIdType
	WEth_GeneralTypes.h	WEthTrcv_SetChanTxParamIdType
	WEth_GeneralTypes.h	WEthTrcv_SetRadioParamIdType
	WEth_GeneralTypes.h	WEthTrcv_TxPwrLvlType

⌋

8.3 Type definitions

8.3.1 EthIf_ConfigType

[SWS_EthIf_00149] Definition of datatype EthIf_ConfigType ⌈

Name	EthIf_ConfigType
Kind	Structure
Description	Implementation specific structure of the post build configuration
Available via	EthIf.h

⌋

8.3.2 EthIf_SwitchPortGroupIdxType

[SWS_EthIf_91101] Definition of datatype EthIf_SwitchPortGroupIdxType ⌈

Name	EthIf_SwitchPortGroupIdxType		
Kind	Type		
Derived from	uint8		
Range	0..255	–	–
Description	Data Type that represents the Ethernet interface switch port group index. The index is zero based and unique for every configured switch port group.		
Available via	EthIf.h		

⌋

8.3.3 EthIf_MeasurementIdxType

[SWS_EthIf_91010] Definition of datatype EthIf_MeasurementIdxType ⌈

Name	EthIf_MeasurementIdxType		
Kind	Type		
Derived from	uint8		
Range	ETHIF_MEAS_DROP_CRTLIDX	0x01	Measurement index of dropped datagrams caused by invalid Crtl Idx/VLAN
	ETHIF_MEAS_RESERVED_1	0x02-0x7F	reserved by AUTOSAR
	ETHIF_MEAS_RESERVED_2	0x80-0xEF	Vendor specific range
	ETHIF_MEAS_RESERVED_3	0xF0-0xFE	reserved by AUTOSAR (future use)
	ETHIF_MEAS_ALL	0xFF	represents all measurement indexes
Description	Index to select specific measurement data		
Available via	EthIf.h		

⌋

8.3.4 EthIf_SignalQualityResultType

[SWS_EthIf_91057] Definition of datatype EthIf_SignalQualityResultType ⌈

Name	EthIf_SignalQualityResultType	
Kind	Structure	
Elements	HighestSignalQuality	
	Type	uint32
	Comment	the highest signal quality of a link since last clear
	LowestSignalQuality	
	Type	uint32
	Comment	the lowest link signal quality of a link since last clear
	ActualSignalQuality	
	Type	uint32
	Comment	the actual signal quality
Description	–	
Available via	EthIf.h	

⌋

8.4 Function definitions

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

Note: All functions in this chapter requires previous initialization (`EthIf_Init`), except the following ones: `EthIf_Init`, `EthIf_GetVersionInfo`

8.4.1 Driver

8.4.1.1 EthIf_SetControllerMode

[SWS_EthIf_00034] Definition of API function EthIf_SetControllerMode ⌈

Service Name	EthIf_SetControllerMode	
Syntax	`Std_ReturnType EthIf_SetControllerMode ( uint8 CtrlIdx, Eth_ModeType CtrlMode )`	
Service ID [hex]	0x03	
Sync/Async	Asynchronous	
Reentrancy	Non Reentrant	
Parameters (in)	CtrlIdx	Index of the Ethernet controller within the context of the Ethernet Interface
	CtrlMode	ETH_MODE_DOWN: disable the controller ETH_MODE_ACTIVE: enable the controller ETH_MODE_ACTIVE_WITH_WAKEUP_REQUEST: enable the controller and request a wake-up on the network. ETH_MODE_TX_OFFLINE: disable transmission handling in Eth If. Please note, the according Ethernet controller is not affected
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: success `E_NOT_OK`: controller mode could not be changed
Description	Enables / disables the indexed controller	
Available via	EthIf.h	

⌋

Note: Further requirements regarding the call of `EthIf_SetControllerMode` are described in chapter 7.1.10.2 and 7.1.11.

8.4.1.2 EthIf_GetControllerMode

[SWS_EthIf_00039] Definition of API function EthIf_GetControllerMode ⌈

Service Name	EthIf_GetControllerMode	
Syntax	`Std_ReturnType EthIf_GetControllerMode ( uint8 CtrlIdx, Eth_ModeType* CtrlModePtr )`	
Service ID [hex]	0x04	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	CtrlIdx	Index of the Ethernet controller within the context of the Ethernet Interface
Parameters (inout)	None	
Parameters (out)	CtrlModePtr	ETH_MODE_DOWN: the controller is disabled ETH_MODE_ ACTIVE: the controller is enabled

▽

△

Return value	Std_ReturnType	E_OK: success E_NOT_OK: controller could not be initialized
Description	Obtains the state of the indexed controller	
Available via	EthIf.h	

⌋

[SWS_EthIf_00040] ⌈The function EthIf_GetControllerMode shall forward the call to function <EthDrv>_GetControllerMode of the corresponding Ethernet Controller Driver EthIfPhysControllerIdx.⌋

8.4.1.3 EthIf_ReadMii

[SWS_EthIf_91239] Definition of API function EthIf_ReadMii ⌈

Service Name	EthIf_ReadMii	
Syntax	`Std_ReturnType EthIf_ReadMii ( uint8 CtrlIdx, uint8 TrcvIdx, uint8 RegIdx, uint16* RegValPtr )`	
Service ID [hex]	0xa1	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different CtrlIdxs. Non reentrant for the same CtrlIdx.	
Parameters (in)	CtrlIdx	Index of the controller.
	TrcvIdx	Index of the transceiver on the SMI.
	RegIdx	Index of the transceiver register on the SMI.
Parameters (inout)	None	
Parameters (out)	RegValPtr	Filled with the register content of the indexed register.
Return value	Std_ReturnType	E_OK: Service accepted E_NOT_OK: Service denied
Description	Reads a transceiver register.	
Available via	EthIf.h	

⌋

[SWS_EthIf_00706] Forwarding a call of EthIf_ReadMii to Eth_ReadMii ⌈The function EthIf_ReadMii shall forward the call to function <EthDrv>_ReadMii of the corresponding Ethernet Controller Driver EthIfPhysControllerIdx.⌋

8.4.1.4 EthIf_WriteMii

[SWS_EthIf_91240] Definition of API function EthIf_WriteMii ⌈

Service Name	EthIf_WriteMii	
Syntax	`Std_ReturnType EthIf_WriteMii ( uint8 CtrlIdx, uint8 TrcvIdx, uint8 RegIdx, uint16 RegVal )`	
Service ID [hex]	0xa0	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different CtrlIdxs. Non reentrant for the same CtrlIdx.	
Parameters (in)	CtrlIdx	Index of the controller.
	TrcvIdx	Index of the transceiver on the SMI.
	RegIdx	Index of the transceiver register on the SMI.
	RegVal	Value to be written into the indexed register.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: Service accepted `E_NOT_OK`: Service denied
Description	Writes a transceiver register.	
Available via	EthIf.h	

⌋

[SWS_EthIf_00707] Forwarding a call of EthIf_WriteMii to Eth_WriteMii ⌈The function `EthIf_WriteMii` shall forward the call to function `<EthDrv>_WriteMii` of the corresponding Ethernet Controller Driver `EthIfPhysControllerIdx`.⌋

8.4.1.5 EthIf_ReadMmd

[SWS_EthIf_91241] Definition of API function EthIf_ReadMmd ⌈

Service Name	EthIf_ReadMmd	
Syntax	`Std_ReturnType EthIf_ReadMmd ( uint8 CtrlIdx, uint8 TrcvIdx, uint8 Mmd, uint16 RegIdx, uint16* RegValPtr )`	
Service ID [hex]	0xa3	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different CtrlIdx, non reentrant for same CtrlIdx	
Parameters (in)	CtrlIdx	Index of the controller.
	TrcvIdx	Index of the transceiver on the SMI.
	Mmd	MDIO Manageable Device.
	RegIdx	Index of the transceiver register on the SMI.

▽

△

Parameters (inout)	None	
Parameters (out)	RegValPtr	Filled with the register content of the indexed register.
Return value	Std_ReturnType	`E_OK`: Service accepted `E_NOT_OK`: Service denied
Description	Reads a transceiver register using Clause 45 access if supported by hardware or implements a Clause 45 access using Clause 22 operations.	
Available via	EthIf.h	

⌋

[SWS_EthIf_00708] Forwarding a call of EthIf_ReadMmd to Eth_ReadMmd ⌈The function `EthIf_ReadMmd` shall forward the call to function `<EthDrv>_ReadMmd` of the corresponding Ethernet Controller Driver `EthIfPhysControllerIdx`.⌋

8.4.1.6 EthIf_WriteMmd

[SWS_EthIf_91242] Definition of API function EthIf_WriteMmd ⌈

Service Name	EthIf_WriteMmd	
Syntax	`Std_ReturnType EthIf_WriteMmd ( uint8 CtrlIdx, uint8 TrcvIdx, uint8 Mmd, uint16 RegIdx, uint16 RegVal )`	
Service ID [hex]	0xa2	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different CtrlIdx, non reentrant for same CtrlIdx.	
Parameters (in)	CtrlIdx	Index of the controller.
	TrcvIdx	Index of the transceiver on the SMI.
	Mmd	MDIO Manageable Device.
	RegIdx	Index of the transceiver register on the SMI.
	RegVal	Value to be written to the given address.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: Service accepted `E_NOT_OK`: Service denied
Description	Writes a transceiver register using Clause 45 access or implements a Clause 45 access using Clause 22 operations.	
Available via	EthIf.h	

⌋

[SWS_EthIf_00709] Forwarding a call of EthIf_WriteMmd to Eth_WriteMmd ⌈The function `EthIf_WriteMmd` shall forward the call to function `<EthDrv>_WriteMmd` of the corresponding Ethernet Controller Driver `EthIfPhysControllerIdx`.⌋

8.4.1.7 EthIf_GetPhysAddr

[SWS_EthIf_00061] Definition of API function EthIf_GetPhysAddr ⌈

Service Name	EthIf_GetPhysAddr	
Syntax	`void EthIf_GetPhysAddr ( uint8 CtrlIdx, uint8* PhysAddrPtr )`	
Service ID [hex]	0x08	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	CtrlIdx	Index of the Ethernet controller within the context of the Ethernet Interface
Parameters (inout)	None	
Parameters (out)	PhysAddrPtr	Physical source address (MAC address) in network byte order.
Return value	None	
Description	Obtains the physical source address used by the indexed controller	
Available via	EthIf.h	

⌋

[SWS_EthIf_00062] ⌈The function `EthIf_GetPhysAddr` shall forward the call to the respective Ethernet Controller Driver.⌋

8.4.1.8 EthIf_SetPhysAddr

[SWS_EthIf_00132] Definition of API function EthIf_SetPhysAddr ⌈

Service Name	EthIf_SetPhysAddr	
Syntax	`void EthIf_SetPhysAddr ( uint8 CtrlIdx, const uint8* PhysAddrPtr )`	
Service ID [hex]	0x0d	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant for the same CtrlIdx, reentrant for different	
Parameters (in)	CtrlIdx	Index of the Ethernet controller within the context of the Ethernet Interface.
	PhysAddrPtr	Pointer to memory containing the physical source address (MAC address) in network byte order.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Sets the physical source address used by the indexed controller.	
Available via	EthIf.h	

⌋

8.4.1.9 EthIf_UpdatePhysAddrFilter

[SWS_EthIf_00139] Definition of API function EthIf_UpdatePhysAddrFilter ⌈

Service Name	EthIf_UpdatePhysAddrFilter	
Syntax	`Std_ReturnType EthIf_UpdatePhysAddrFilter ( uint8 CtrlIdx, const uint8* PhysAddrPtr, Eth_FilterActionType Action )`	
Service ID [hex]	0x0c	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant for the same CtrlIdx, reentrant for different	
Parameters (in)	CtrlIdx	Index of the Ethernet controller within the context of the Ethernet Interface.
	PhysAddrPtr	Pointer to memory containing the physical destination address (MAC address) in network byte order. This is the multicast destination address of the layer 2 Ethernet packet.
	Action	Add or remove the address from the Ethernet controllers filter.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: filter was successfully changed `E_NOT_OK`: filter could not be changed
Description	Update the physical source address to/from the indexed controller filter. If the Ethernet Controller is not capable to do the filtering or managing same multicast address of several virtual controllers, the Ethernet Controller software has to handle it.	
Available via	EthIf.h	

⌋

[SWS_EthIf_00140] ⌈The function `EthIf_SetPhysAddrFilter` shall call `Eth_UpdatePhysAddrFilter` with the referenced physical Ethernet Controller index and VlanId. If no VlanId is used (i.e. the affected EthIfContoller has no EthIfVlanId configured) the function shall be called with VlanId 0xFFFF.⌋

8.4.1.10 EthIf_GetPortMacAddr

[SWS_EthIf_00190] Definition of API function EthIf_GetPortMacAddr ⌈

Service Name	EthIf_GetPortMacAddr	
Syntax	`Std_ReturnType EthIf_GetPortMacAddr ( const uint8* MacAddrPtr, uint8* SwitchIdxPtr, uint8* PortIdxPtr )`	
Service ID [hex]	0x28	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	MacAddrPtr	MAC-address for which a switch port is searched over which the node with this MAC-address can be reached.
Parameters (inout)	None	

▽

△

Parameters (out)	SwitchIdxPtr	Pointer to the switch index
	PortIdxPtr	Pointer to the port index
Return value	Std_ReturnType	E_OK: success E_NOT_OK: an error occurred, e.g. multiple ports were found
Description	Obtains the port over which this MAC-address can be reached	
Available via	EthIf.h	

⌋

[SWS_EthIf_00191] ⌈The function EthIf_GetPortMacAddr shall return the switch and port index over which the given MAC-address is reachable. If multiple or no ports are possible, this API call will return E_NOT_OK. EthSwt_GetPortMacAddr will be called for all Ethernet Switch drivers.⌋

[SWS_EthIf_00192] ⌈The function shall be pre compile time configurable On/Off by the configuration parameter: EthIfGetPortMacAddrVlanApi.⌋

8.4.1.11 EthIf_GetPortMacAddrVlan

[SWS_EthIf_91140] Definition of API function EthIf_GetPortMacAddrVlan ⌈

Service Name	EthIf_GetPortMacAddrVlan	
Syntax	`Std_ReturnType EthIf_GetPortMacAddrVlan ( uint8 SwitchIdx, const uint8* MacAddrPtr, const uint16* VlanIdPtr, uint32* PortBitMapPtr )`	
Service ID [hex]	0x9d	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	SwitchIdx	Index of the Ethernet switch within the context of the Ethernet Switch Driver
	MacAddrPtr	MAC-address which is requested to look-up the assignment to an Ethernet switch port
	VlanIdPtr	VlanId which is requested to look-up the assignment to an Ethernet switch port
Parameters (inout)	None	
Parameters (out)	PortBitMapPtr	Returns a pointer to an Ethernet switch port bit map, where the requested MAC-address with respect to the given VLAN-ID is available
Return value	Std_ReturnType	E_OK: success E_NOT_OK: request could not be successfully finalized, due to several possible reasons (e.g. requested Ethernet switch addressed with switchIdx is not valid or inactive)

▽

△

Description	Obtains a Ethernet switch port bit map, where the given MAC-address with respect to the given VLAN-ID is assigned to. The return argurment PortBitMapPtr points to uint32 value which shall be handled as Ethernet switch port bit map. Each bit of the Ethernet switch port bit map represents a EthSwtPortIdx, where the least significant bit (bit 0) represents EthSwichtPortIdx 0 and most signification bit (bit 32) represents EthSwichtPortIdx 31 (e.g. 0x0001 == EthSwicht PortIdx 0 is set; 0x8005 == EthSwichtPortIdx 0, 2 and 31 are set
Available via	EthIf.h

⌋

[SWS_EthIf_00659] Behaviour if function is called

Upstream requirements: SRS_Eth_00182

⌈The function `EthIf_GetPortMacAddrVlan` shall forward the call to the EthSwt driver by calling `EthSwt_GetPortMacAddrVlan`.⌋

[SWS_EthIf_00660] Pre compile configuration switch

Upstream requirements: SRS_Eth_00182

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

8.4.1.12 EthIf_GetArlTable

[SWS_EthIf_00196] Definition of API function EthIf_GetArlTable ⌈

Service Name	EthIf_GetArlTable	
Syntax	`Std_ReturnType EthIf_GetArlTable ( uint8 switchIdx, uint16* numberOfElements, Eth_MacVlanType* arlTableListPointer )`	
Service ID [hex]	0x29	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	switchIdx	Index of the switch within the context of the Ethernet Switch Driver
Parameters (inout)	numberOfElements	In: Maximum number of elements which can be written into the arlTable Out: Number of elements which are currently available in the EthSwitch module.
Parameters (out)	arlTableListPointer	Returns a pointer to the memory where the ARL table of the switch consisting of a list of structs with MAC-address, VLAN-ID and port shall be stored.
Return value	Std_ReturnType	`E_OK`: success `E_NOT_OK`: requested switchIdx is not valid or inactive
Description	Obtains the address resolution table of a switch and copies the list into a user provided buffer. The function will copy all or numberOfElements into the output list. If input value of numberOf Elements is 0 the function will not copy any data but only return the number of valid entries in the cache. arlTableListPointer may be NULL_PTR in this case.	
Available via	EthIf.h	

⌋

[SWS_EthIf_00197] ⌈The function EthIf_GetArlTable shall return a list of structs with MAC-address, VLAN-ID and port for the indexed switch.⌋

[SWS_EthIf_00254] ⌈The function EthIf_GetArlTable shall forward the call to function EthSwt_GetArlTable of the respective Ethernet Switch Driver.⌋

[SWS_EthIf_00198] ⌈The function shall be pre compile time configurable On/Off by the configuration parameter: EthIfGetArlTable.⌋

8.4.1.13 EthIf_Transmit

[SWS_EthIf_00075] Definition of API function EthIf_Transmit ⌈

Service Name	EthIf_Transmit	
Syntax	`Std_ReturnType EthIf_Transmit ( PduIdType TxPduId, const PduInfoType* PduInfoPtr )`	
Service ID [hex]	0x1d	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in)	TxPduId	Identifier of the PDU to be transmitted
	PduInfoPtr	Length of and pointer to the PDU data and pointer to MetaData.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: Transmit request has been accepted. E_NOT_OK: Transmit request has not been accepted.
Description	Requests transmission of a PDU.	
Available via	EthIf.h	

⌋

[SWS_EthIf_00250] ⌈If CtrlIdx refers to an EthIfCtrl where an EthIfVlanID is configured, the parameters FrameType is not used, and 0x8100 is provided to <EthDrv>_Transmit instead.⌋

[SWS_EthIf_00076] ⌈If the latest accepted controller mode is equal to ETH_MODE_ACTIVE or ETH_MODE_ACTIVE_WITH_WAKEUP_REQUEST for the given EthIfController, then the function EthIf_Transmit shall forward the call to the respective Ethernet Controller Driver. Otherwise the function shall reject the request for a transmission and return with E_NOT_OK.⌋

8.4.1.14 EthIf_EthGetSpiStatus

[SWS_EthIf_91022] Definition of API function EthIf_EthGetSpiStatus

Status: DRAFT

⌈

Service Name	EthIf_EthGetSpiStatus (draft)	
Syntax	`Std_ReturnType EthIf_EthGetSpiStatus ( uint8* CtrlIdx, Eth_SpiStatusType* SpiStatusPtr )`	
Service ID [hex]	0x6a	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	CtrlIdx	Index of the controller within the context of the Ethernet Interface.
Parameters (inout)	None	
Parameters (out)	SpiStatusPtr	Status of the SPI interface
Return value	Std_ReturnType	`E_OK`: success. `E_NOT_OK`: Controller request has not been accepted.
Description	When MACPHY controller are used, obtains the SPI interface status. **Tags:** atp.Status=draft	
Available via	EthIf.h	

⌋

[SWS_EthIf_00505]

Status: DRAFT

⌈The function `EthIf_EthGetSpiStatus` shall forward the call to function `<EthDrv>_GetSpiStatus` of the corresponding Ethernet Driver (`CtrlIdx`).⌋

8.4.2 Firewall

8.4.2.1 EthIf_GetStreamStatistics

[SWS_EthIf_91027] Definition of API function EthIf_GetStreamStatistics

Status: DRAFT
Upstream requirements: FO_RS_Fw_00011

⌈

Service Name	EthIf_GetStreamStatistics (draft)	
Syntax	`void EthIf_GetStreamStatistics ( uint8 SwitchIdx )`	
Service ID [hex]	0x91	
Sync/Async	Synchronous	
Reentrancy	Reentrant	

▽

△

Parameters (in)	SwitchIdx	Index of the switch within the context of the Ethernet Switch Driver
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Requests the statistics (bucket counter values) of an Ethernet switch of all configured streams. **Tags:** atp.Status=draft	
Available via	EthIf.h	

⌋

8.4.2.2 EthIf_SetStreamState

[SWS_EthIf_91025] Definition of API function EthIf_SetStreamState

Status: DRAFT
Upstream requirements: FO_RS_Fw_00011

⌈

Service Name	EthIf_SetStreamState (draft)	

Syntax	`void EthIf_SetStreamState (`
	`  uint8 SwitchIdx,`
	`  uint8 StreamHandleIdxPtr,`
	`  boolean StreamActivityStatus`
	`)`	

Service ID [hex]	0x92	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	SwitchIdx	Index of the switch within the context of the Ethernet Switch Driver
	StreamHandleIdxPtr	Pointer to the StreamHandleIdx for which the status shall be set
	StreamActivityStatus	Activity status of the StreamHandleIdx (True = active, False = inactive) to be set
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	This function is called by the Firewall module to control the activity status of a stream in the Ethernet switch. **Tags:** atp.Status=draft	
Available via	EthIf.h	

⌋

8.4.3 General

8.4.3.1 EthIf_Init

[SWS_EthIf_00024] Definition of API function EthIf_Init ⌈

Service Name	EthIf_Init	
Syntax	`void EthIf_Init ( const EthIf_ConfigType* CfgPtr )`	
Service ID [hex]	0x01	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	CfgPtr	Points to the implementation specific structure
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Initializes the Ethernet Interface	
Available via	EthIf.h	

⌋

[SWS_EthIf_00025] ⌈The function shall store the access to the configuration structure for subsequent API calls.⌋

[SWS_EthIf_00114] ⌈The function shall change the state of the component from uninitialized to initialized.⌋

[SWS_EthIf_00116] ⌈If development error detection is enabled: the function shall check the parameter `CfgPtr` for containing a valid configuration. If the check fails, the function shall raise the development error `ETHIF_E_INIT_FAILED`.⌋

8.4.3.2 EthIf_GetCtrlIdxList

[SWS_EthIf_91053] Definition of API function EthIf_GetCtrlIdxList ⌈

Service Name	EthIf_GetCtrlIdxList	
Syntax	`Std_ReturnType EthIf_GetCtrlIdxList ( uint8* NumberOfCtrlIdx, uint8* CtrlIdxListPtr )`	
Service ID [hex]	0x44	
Sync/Async	Asynchronous	
Reentrancy	Non Reentrant	
Parameters (in)	None	
Parameters (inout)	NumberOfCtrlIdx	in: maximum number of controllers in CtrlIdxListPtr, 0 to return the number of controllers but without filling CtrlIdxListPtr. out: number of active controllers.
Parameters (out)	CtrlIdxListPtr	List of active controller indexes

▽

△

Return value	Std_ReturnType	`E_OK`: success `E_NOT_OK`: failure
Description	Returns the number and index of all active Ethernet controllers.	
Available via	EthIf.h	

⌋

[SWS_EthIf_00298] ⌈The optional `EthIf_GetCtrlIdxList` API shall return only the `NumberOfCtrlIdx` which are active.⌋

8.4.3.3 EthIf_GetVlanId

[SWS_EthIf_91052] Definition of API function EthIf_GetVlanId ⌈

Service Name	EthIf_GetVlanId	
Syntax	`Std_ReturnType EthIf_GetVlanId ( uint8 CtrlIdx, uint16* VlanIdPtr )`	
Service ID [hex]	0x43	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	CtrlIdx	Index of the Ethernet controller within the context of the Ethernet Interface
Parameters (inout)	None	
Parameters (out)	VlanIdPtr	Pointer to store the VLAN identifier (VID) of the Ethernet controller. 0 if the the Ethernet controller represents no virtual network (VLAN).
Return value	Std_ReturnType	`E_OK`: success `E_NOT_OK`: failure
Description	Returns the VLAN identifier of the requested Ethernet controller.	
Available via	EthIf.h	

⌋

[SWS_EthIf_00301] ⌈The optional `EthIf_GetVlanId` API shall return the VlanId of the requested `CtrlIdx`.⌋

8.4.3.4 EthIf_GetAndResetMeasurementData

[SWS_EthIf_91011] Definition of API function EthIf_GetAndResetMeasurement Data ⌈

Service Name	EthIf_GetAndResetMeasurementData	
Syntax	`Std_ReturnType EthIf_GetAndResetMeasurementData ( EthIf_MeasurementIdxType MeasurementIdx, boolean MeasurementResetNeeded, uint32* MeasurementDataPtr )`	
Service ID [hex]	0x45	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	MeasurementIdx	Data index of measurement data
	MeasurementReset Needed	Flag to trigger a reset of the measurement data
Parameters (inout)	None	
Parameters (out)	MeasurementDataPtr	Reference to data buffer, where to copy measurement data
Return value	Std_ReturnType	E_OK: successful E_NOT_OK: failed
Description	Allows to read and reset detailed measurement data for diagnostic purposes. Get all MeasurementIdx's at once is not supported. ETHIF_MEAS_ALL shall only be used to reset all MeasurementIdx's at once. A NULL_PTR shall be provided for MeasurementDataPtr in this case.	
Available via	EthIf.h	

⌋

[SWS_EthIf_00308] ⌈EthIf_GetAndResetMeasurementData shall return measurement data for selected measurement index.⌋

[SWS_EthIf_00309] ⌈For measurement index ETHIF_MEAS_DROP_CRTLIDX the function shall return the number of all dropped datagrams, caused by invalid CrtlIdx/VLAN. If the VLAN is not enabled, all received VLAN tagged datagrams are invalid and shall be counted also.⌋

[SWS_EthIf_00310] ⌈The function shall return E_NOT_OK if the requested measurement index is not supported.⌋

[SWS_EthIf_00312] ⌈The function shall reset all existing measurement data to 0, if MeasurementResetNeeded is true and measurement index is set to ETHIF_MEAS_ALL.⌋

[SWS_EthIf_00313] ⌈All measurement data which counts data shall not overrun.⌋

[SWS_EthIf_00314] ⌈The function shall accept NULL_PTR. In this case the measurement data shall not be copied.⌋

[SWS_EthIf_00316] ⌈The function shall be pre compile time configurable On/Off by the configuration parameter: EthIfGetAndResetMeasurementDataApi.⌋

[SWS_EthIf_00317] ⌈If the VLAN is not active the Ethernet Interface shall increment the corresponding measurement data and filter the message.⌋

8.4.3.5 EthIf_VerifyConfig

[SWS_EthIf_91012] Definition of API function EthIf_VerifyConfig ⌈

Service Name	EthIf_VerifyConfig	

Syntax	`Std_ReturnType EthIf_VerifyConfig (`
	`  uint8 SwitchIdx,`
	`  boolean* Result`
	`)`	

Service ID [hex]	0x40	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	SwitchIdx	Index of the switch within the context of the Ethernet Interface
Parameters (inout)	None	
Parameters (out)	Result	Result of verification, TRUE: configureation verified ok, FALSE: configuraton values found corrupted
Return value	Std_ReturnType	`E_OK`: Configuration verificaton succeeded, `E_NOT_OK`: Configuration verification not succeeded.
Description	Forwarded to EthSwt_VerifyConfig. EthSwt_VerifyConfig verifies the Switch Configuration depending on the HW-Architecture, HW-capability and the intended accuracy of this verification.	
Available via	EthIf.h	

⌋

[SWS_EthIf_00305] ⌈The function shall be compile time configurable On/Off by the configuration parameter: `EthIfVerifyConfigApi`.⌋

8.4.3.6 EthIf_SetForwardingMode

[SWS_EthIf_91013] Definition of API function EthIf_SetForwardingMode ⌈

Service Name	EthIf_SetForwardingMode	

Syntax	`Std_ReturnType EthIf_SetForwardingMode (`
	`  uint8 SwitchIdx,`
	`  boolean mode`
	`)`	

Service ID [hex]	0x41	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	SwitchIdx	Index of the switch within the context of the Ethernet Switch Driver
	mode	True Forwarding enabled, False Forwarding disabled
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: stopping of frame forwarding succeeded, `E_NOT_OK`: stopping of frame forwarding not succeeded.

▽

△

Description	Verifies the Switch Configuration. If Configuration is not valid, Switch is reconfigured.
Available via	EthIf.h

⌋

[SWS_EthIf_00307] ⌈The function shall be compile time configurable On/Off by the configuration parameter: `EthIfSetForwardingModeApi`.⌋

8.4.3.7 EthIf_ClearTrcvSignalQuality

[SWS_EthIf_91059] Definition of API function EthIf_ClearTrcvSignalQuality ⌈

Service Name	EthIf_ClearTrcvSignalQuality	
Syntax	`Std_ReturnType EthIf_ClearTrcvSignalQuality ( uint8 TrcvIdx )`	
Service ID [hex]	0x19	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different TrcvIdx. Non reentrant for the same TrcvIdx.	
Parameters (in)	TrcvIdx	Index of the transceiver within the context of the Ethernet Interface
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: The signal quality cleared successfully `E_NOT_OK`: The signal quality cleared not successfully
Description	Clear the stored signal quality of the link of the given Ethernet transceiver	
Available via	EthIf.h	

⌋

[SWS_EthIf_00400] ⌈The function `EthIf_ClearTrcvSignalQuality` shall clear the stored signal quality values (see `EthIf_SignalQualityResultType`) of the EthIfTransceiver given by `TrcvIdx`.⌋

8.4.3.8 EthIf_ClearSwitchPortSignalQuality

[SWS_EthIf_91060] Definition of API function EthIf_ClearSwitchPortSignalQuality ⌈

Service Name	EthIf_ClearSwitchPortSignalQuality
Syntax	`Std_ReturnType EthIf_ClearSwitchPortSignalQuality ( uint8 SwitchIdx, uint8 SwitchPortIdx )`
Service ID [hex]	0x1b
Sync/Async	Synchronous

▽

△

Reentrancy	Reentrant for different Ethernet switch indexes and Ethernet Switch port indexes. Non reentrant for the same SwitchPortIdx.	
Parameters (in)	SwitchIdx	Index of the Ethernet switch within the context of the Ethernet Interface
	SwitchPortIdx	Index of the Ethernet switch port within the context of the Ethernet Interface
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: The signal quality cleared successfully `E_NOT_OK`: The signal quality cleared not successfully
Description	Clear the stored signal quality of the link of the given Ethernet switch port	
Available via	EthIf.h	

⌋

[SWS_EthIf_00404] ⌈The function `EthIf_ClearSwitchPortSignalQuality` shall clear the stored signal quality values (see `EthIf_SignalQualityResult-Type`) of the EthSwtPort given by `SwitchIdx` and `SwitchPortIdx`.⌋

8.4.3.9 EthIf_ReleaseRxBuffer

[SWS_EthIf_91138] Definition of API function EthIf_ReleaseRxBuffer

Status: DRAFT
Upstream requirements: SRS_Eth_00169

⌈

Service Name	EthIf_ReleaseRxBuffer (draft)	

Syntax	`void EthIf_ReleaseRxBuffer (`
	`  PduIdType RxPduId`
	`)`	

Service ID [hex]	0x9b	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different PduIds. Non reentrant for the same PduId	
Parameters (in)	RxPduId	Identifier of the received PDU.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Indication from the upper layer to release the lower layer reception buffer. **Tags:** atp.Status=draft	
Available via	EthIf.h	

⌋

8.4.3.10 EthIf_GetVersionInfo

[SWS_EthIf_00082] Definition of API function EthIf_GetVersionInfo ⌈

Service Name	EthIf_GetVersionInfo	
Syntax	`void EthIf_GetVersionInfo ( Std_VersionInfoType* VersionInfoPtr )`	
Service ID [hex]	0x0b	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	None	
Parameters (inout)	None	
Parameters (out)	VersionInfoPtr	Version information of this module
Return value	None	
Description	Returns the version information of this module	
Available via	EthIf.h	

⌋

8.4.4 MacSec

8.4.4.1 EthIf_SwitchMacSecUpdateSecY

[SWS_EthIf_91219] Definition of API function EthIf_SwitchMacSecUpdateSecY

Status: DRAFT

⌈

Service Name	EthIf_SwitchMacSecUpdateSecY (draft)	
Syntax	`Std_ReturnType EthIf_SwitchMacSecUpdateSecY ( const EthSwt_MgmtInfoType* MgmtInfoPtr, const Mka_MacSecConfigType* MACsecCfgPtr, uint64 TxSci )`	
Service ID [hex]	0x6d	
Sync/Async	Asynchronous	
Reentrancy	Reentrant for different MgmtInfoPtr, Non reentrant for the same MgmtInfoPtr	
Parameters (in)	MgmtInfoPtr	Pointer to the management information within the context of an Ethernet Switch Driver. SwitchIdx in context of the EthIf (EthIf Switch/EthIfSwitchIdx), PortIdx in context of EthSwt (EthSwtPort/ EthSwtPortIdx).
	MACsecCfgPtr	Pointer to the structure to configure a MACsec Entity (SecY)
	TxSci	Secure Channel Identifier for the MACsec's Transmission Secure channel
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: The request has been accepted `E_NOT_OK`: The request has not been accepted

▽

△

Description	Requests the Ethernet Switch to update the SecY/PAC of the the provided port with the provided parameters. A Transmission Secure Channel with the provided SCI shall be configured during the first call. A pointer to a MACsec Basic Parameters Configuration file shall be provided to create the Secure Channel. **Tags:** atp.Status=draft
Available via	EthIf.h

⌋

8.4.4.2 EthIf_MacSecUpdateSecY

[SWS_EthIf_91215] Definition of API function EthIf_MacSecUpdateSecY

Status: DRAFT

⌈

Service Name	EthIf_MacSecUpdateSecY (draft)	

Syntax	`Std_ReturnType EthIf_MacSecUpdateSecY (`
	`  uint8 CtrlIdx,`
	`  const Mka_MacSecConfigType* MACsecCfgPtr,`
	`  uint64 TxSci`
	`)`	

Service ID [hex]	0x88	
Sync/Async	Asynchronous	
Reentrancy	Reentrant for different CtrlIdx, Non reentrant for the same CtrlIdx	
Parameters (in)	CtrlIdx	Index of the Ethernet controller within the context of the Ethernet Interface
	MACsecCfgPtr	Pointer to the structure to configure a MACsec Entity (SecY)
	TxSci	Secure Channel Identifier for the MACsec's Transmission Secure channel
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: The request has been accepted `E_NOT_OK`: The request has not been accepted
Description	Requests the Ethernet Inferface (MACsec per SW) or the Ethernet Transceiver to update the SecY/PAC of the PHY with the provided parameters. A Transmission Secure Channel with the provided SCI shall be configured during the first call. A pointer to a MACsec Basic Parameters Configuration file shall be provided to create the Secure Channel. **Tags:** atp.Status=draft	
Available via	EthIf.h	

⌋

8.4.4.3 EthIf_SwitchMacSecInitRxSc

[SWS_EthIf_91220] Definition of API function EthIf_SwitchMacSecInitRxSc

Status: DRAFT

⌈

Service Name	EthIf_SwitchMacSecInitRxSc (draft)	

Syntax	`Std_ReturnType EthIf_SwitchMacSecInitRxSc (`
	`  const EthSwt_MgmtInfoType* MgmtInfoPtr,`
	`  uint64 Sci`
	`)`	

Service ID [hex]	0x6e	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different MgmtInfoPtr, Non reentrant for the same MgmtInfoPtr	
Parameters (in)	MgmtInfoPtr	Pointer to the management information within the context of an Ethernet Switch Driver. SwitchIdx in context of the EthIf (EthIf Switch/EthIfSwitchIdx), PortIdx in context of EthSwt (EthSwtPort/ EthSwtPortIdx).
	Sci	Secure Channel Identifier for the MACsec's Reception Secure channel
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: The request has been accepted `E_NOT_OK`: The request has not been accepted
Description	Requests the Ethernet Switch Driver to configure a Reception Secure Channel for the given Secure Channel Identifier. **Tags:** atp.Status=draft	
Available via	EthIf.h	

⌋

8.4.4.4 EthIf_MacSecInitRxSc

[SWS_EthIf_91211] Definition of API function EthIf_MacSecInitRxSc

Status: DRAFT

⌈

Service Name	EthIf_MacSecInitRxSc (draft)	

Syntax	`Std_ReturnType EthIf_MacSecInitRxSc (`
	`  uint8 CtrlIdx,`
	`  uint64 Sci`
	`)`	

Service ID [hex]	0x87	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different CtrlIdx, Non reentrant for the same CtrlIdx	
Parameters (in)	CtrlIdx	Index of the Ethernet controller within the context of the Ethernet Interface
	Sci	Secure Channel Identifier for the MACsec's Reception Secure channel
Parameters (inout)	None	

▽

△

Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: The request has been accepted `E_NOT_OK`: The request has not been accepted
Description	Requests the Ethernet Inferface (MACsec per SW) or the Ethernet Transceiver Driver to configure a Reception Secure Channel for the given Secure Channel Identifier. **Tags:** atp.Status=draft	
Available via	EthIf.h	

⌋

8.4.4.5 EthIf_SwitchMacSecResetRxSc

[SWS_EthIf_91221] Definition of API function EthIf_SwitchMacSecResetRxSc

Status: DRAFT

⌈

Service Name	EthIf_SwitchMacSecResetRxSc (draft)	

Syntax	`Std_ReturnType EthIf_SwitchMacSecResetRxSc (`
	`  const EthSwt_MgmtInfoType* MgmtInfoPtr,`
	`  uint64 Sci`
	`)`	

Service ID [hex]	0x6f	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different MgmtInfoPtr, Non reentrant for the same MgmtInfoPtr	
Parameters (in)	MgmtInfoPtr	Pointer to the management information within the context of an Ethernet Switch Driver. SwitchIdx in context of the EthIf (EthIf Switch/EthIfSwitchIdx), PortIdx in context of EthSwt (EthSwtPort/ EthSwtPortIdx).
	Sci	Secure Channel Identifier for the MACsec's Reception Secure channel
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: The request has been accepted `E_NOT_OK`: The request has not been accepted
Description	Requests the Ethernet Switch Driver to reset to default the MACsec values of the Reception Secure Channel for the given Secure Channel Identifier. **Tags:** atp.Status=draft	
Available via	EthIf.h	

⌋

8.4.4.6 EthIf_MacSecResetRxSc

[SWS_EthIf_91213] Definition of API function EthIf_MacSecResetRxSc

Status: DRAFT

⌈

Service Name	EthIf_MacSecResetRxSc (draft)	

Syntax	`Std_ReturnType EthIf_MacSecResetRxSc (`
	`  uint8 CtrlIdx,`
	`  uint64 Sci`
	`)`	

Service ID [hex]	0x86	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different CtrlIdx, Non reentrant for the same CtrlIdx	
Parameters (in)	CtrlIdx	Index of the Ethernet controller within the context of the Ethernet Interface
	Sci	Secure Channel Identifier for the MACsec's Reception Secure channel
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: The request has been accepted `E_NOT_OK`: The request has not been accepted
Description	Requests the Ethernet Inferface (MACsec per SW) or the Ethernet Transceiver Driver to reset to default the MACsec values of the Reception Secure Channel for the given Secure Channel Identifier. **Tags:** atp.Status=draft	
Available via	EthIf.h	

⌋

8.4.4.7 EthIf_SwitchMacSecAddTxSa

[SWS_EthIf_91222] Definition of API function EthIf_SwitchMacSecAddTxSa

Status: DRAFT

⌈

Service Name	EthIf_SwitchMacSecAddTxSa (draft)

Syntax	`Std_ReturnType EthIf_SwitchMacSecAddTxSa (`
	`  const EthSwt_MgmtInfoType* MgmtInfoPtr,`
	`  uint8 An,`
	`  uint64 NextPn,`
	`  uint32 Ssci,`
	`  const Mka_SakKeyPtrType* KeysPtr,`
	`  boolean Active`
	`)`

Service ID [hex]	0x70
Sync/Async	Asynchronous
Reentrancy	Reentrant for different MgmtInfoPtr, Non reentrant for the same MgmtInfoPtr

▽

△

Parameters (in)	MgmtInfoPtr	Pointer to the management information within the context of an Ethernet Switch Driver. SwitchIdx in context of the EthIf (EthIf Switch/EthIfSwitchIdx), PortIdx in context of EthSwt (EthSwtPort/ EthSwtPortIdx).
	An	Association Number to use in the MACsec's transmission secure association
	NextPn	Next accepted Packet Number in the MACsec's transmission secure association
	Ssci	Short Secure Channel Identifiert used in the MACsec's transmission secure association
	KeysPtr	Pointer to the SAKs Key (and needed Key information) to use in the MACsec's transmission secure association
	Active	Boolean to enable/disable the MACsec's transmission secure association
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: The request has been accepted `E_NOT_OK`: The request has not been accepted
Description	Requests the Ethernet Switch Driver to create a Transmission Secure Association in the provided port. The Short Secure Channel Identifier is included to support XPN configurations. **Tags:** atp.Status=draft	
Available via	EthIf.h	

⌋

8.4.4.8 EthIf_MacSecAddTxSa

[SWS_EthIf_91206] Definition of API function EthIf_MacSecAddTxSa

Status: DRAFT

⌈

Service Name	EthIf_MacSecAddTxSa (draft)	
Syntax	`Std_ReturnType EthIf_MacSecAddTxSa ( uint8 CtrlIdx, uint8 An, uint64 NextPn, uint32 Ssci, const Mka_SakKeyPtrType* KeysPtr, boolean Active )`	
Service ID [hex]	0x85	
Sync/Async	Asynchronous	
Reentrancy	Reentrant for different CtrlIdx, Non reentrant for the same CtrlIdx	
Parameters (in)	CtrlIdx	Index of the Ethernet controller within the context of the Ethernet Interface
	An	Association Number to use in the MACsec's transmission secure association
	NextPn	Next accepted Packet Number in the MACsec's transmission secure association
	Ssci	Short Secure Channel Identifiert used in the MACsec's transmission secure association

▽

△

	KeysPtr	Pointer to the SAKs Key (and needed Key information) to use in the MACsec's transmission secure association
	Active	Boolean to enable/disable the MACsec's transmission secure association
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: The request has been accepted `E_NOT_OK`: The request has not been accepted
Description	Requests the Ethernet Inferface (MACsec per SW) or the Ethernet Transceiver Driver to create a Transmission Secure Association in the Transceiver. The Short Secure Channel Identifier is included to support XPN configurations. **Tags:** atp.Status=draft	
Available via	EthIf.h	

⌋

8.4.4.9 EthIf_SwitchMacSecUpdateTxSa

[SWS_EthIf_91225] Definition of API function EthIf_SwitchMacSecUpdateTxSa

Status: DRAFT

⌈

Service Name	EthIf_SwitchMacSecUpdateTxSa (draft)	

Syntax	`Std_ReturnType EthIf_SwitchMacSecUpdateTxSa (`
	`  const EthSwt_MgmtInfoType* MgmtInfoPtr,`
	`  uint8 An,`
	`  uint64 NextPn,`
	`  boolean Active`
	`)`	

Service ID [hex]	0x73	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different MgmtInfoPtr, Non reentrant for the same MgmtInfoPtr	
Parameters (in)	MgmtInfoPtr	Pointer to the management information within the context of an Ethernet Switch Driver. SwitchIdx in context of the EthIf (EthIf Switch/EthIfSwitchIdx), PortIdx in context of EthSwt (EthSwtPort/ EthSwtPortIdx).
	An	Association Number to use in the MACsec's transmission secure association
	NextPn	Next accepted Packet Number in the MACsec's transmission secure association
	Active	Boolean to enable/disable the MACsec's transmission secure association
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: The request has been accepted `E_NOT_OK`: The request has not been accepted
Description	Requests the Ethernet Switch Driver to update the Transmission Secure Association with the given Packet Number. The Active parameter is included to change the specified AN status. **Tags:** atp.Status=draft	
Available via	EthIf.h	

⌋

8.4.4.10 EthIf_MacSecUpdateTxSa

[SWS_EthIf_91216] Definition of API function EthIf_MacSecUpdateTxSa

Status: DRAFT

⌈

Service Name	EthIf_MacSecUpdateTxSa (draft)	

Syntax	`Std_ReturnType EthIf_MacSecUpdateTxSa (`
	`  uint8 CtrlIdx,`
	`  uint8 An,`
	`  uint64 NextPn,`
	`  boolean Active`
	`)`	

Service ID [hex]	0x84	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different CtrlIdx, Non reentrant for the same CtrlIdx	
Parameters (in)	CtrlIdx	Index of the Ethernet controller within the context of the Ethernet Interface
	An	Association Number to use in the MACsec's transmission secure association
	NextPn	Next accepted Packet Number in the MACsec's transmission secure association
	Active	Boolean to enable/disable the MACsec's transmission secure association
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: The request has been accepted `E_NOT_OK`: The request has not been accepted
Description	Requests the Ethernet Inferface (MACsec per SW) or the Ethernet Transceiver Driver to update the Transmission Secure Association with the given Packet Number. The Active parameter is included to change the specified AN status. **Tags:** atp.Status=draft	
Available via	EthIf.h	

⌋

8.4.4.11 EthIf_SwitchMacSecDeleteTxSa

[SWS_EthIf_91226] Definition of API function EthIf_SwitchMacSecDeleteTxSa

Status: DRAFT

⌈

Service Name	EthIf_SwitchMacSecDeleteTxSa (draft)

Syntax	`Std_ReturnType EthIf_SwitchMacSecDeleteTxSa (`
	`  const EthSwt_MgmtInfoType* MgmtInfoPtr,`
	`  uint8 An`
	`)`

Service ID [hex]	0x74
Sync/Async	Synchronous
Reentrancy	Reentrant for different MgmtInfoPtr, Non reentrant for the same MgmtInfoPtr

▽

△

Parameters (in)	MgmtInfoPtr	Pointer to the management information within the context of an Ethernet Switch Driver. SwitchIdx in context of the EthIf (EthIf Switch/EthIfSwitchIdx), PortIdx in context of EthSwt (EthSwtPort/ EthSwtPortIdx).
	An	Association Number to use in the MACsec's transmission secure association
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: The request has been accepted `E_NOT_OK`: The request has not been accepted
Description	Request the Ethernet Switch Driver to remove the Transmission Secure Association identified by the provided Association Number. **Tags:** atp.Status=draft	
Available via	EthIf.h	

⌋

8.4.4.12 EthIf_MacSecDeleteTxSa

[SWS_EthIf_91208] Definition of API function EthIf_MacSecDeleteTxSa

Status: DRAFT

⌈

Service Name	EthIf_MacSecDeleteTxSa (draft)	
Syntax	`Std_ReturnType EthIf_MacSecDeleteTxSa ( uint8 CtrlIdx, uint8 An )`	
Service ID [hex]	0x16	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different CtrlIdx, Non reentrant for the same CtrlIdx	
Parameters (in)	CtrlIdx	Index of the Ethernet controller within the context of the Ethernet Interface
	An	Association Number to use in the MACsec's transmission secure association
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: The request has been accepted `E_NOT_OK`: The request has not been accepted
Description	Request the Ethernet Inferface (MACsec per SW) or the Ethernet Transceiver Driver to remove the Transmission Secure Association identified by the provided Association Number. **Tags:** atp.Status=draft	
Available via	EthIf.h	

⌋

8.4.4.13 EthIf_SwitchMacSecAddRxSa

[SWS_EthIf_91227] Definition of API function EthIf_SwitchMacSecAddRxSa

Status: DRAFT

⌈

Service Name	EthIf_SwitchMacSecAddRxSa (draft)	

Syntax	`Std_ReturnType EthIf_SwitchMacSecAddRxSa (`
	`  const EthSwt_MgmtInfoType* MgmtInfoPtr,`
	`  uint8 An,`
	`  uint64 LowestPn,`
	`  uint32 Ssci,`
	`  const Mka_SakKeyPtrType* KeysPtr,`
	`  boolean Active`
	`)`	

Service ID [hex]	0x75	
Sync/Async	Asynchronous	
Reentrancy	Reentrant for different MgmtInfoPtr, Non reentrant for the same MgmtInfoPtr	
Parameters (in)	MgmtInfoPtr	Pointer to the management information within the context of an Ethernet Switch Driver. SwitchIdx in context of the EthIf (EthIf Switch/EthIfSwitchIdx), PortIdx in context of EthSwt (EthSwtPort/ EthSwtPortIdx).
	An	Association Number to use in the MACsec's reception secure association
	LowestPn	Lowest accepted Packet Number in the MACsec's reception secure association
	Ssci	Short Secure Channel Identifiert used in the MACsec's reception secure association
	KeysPtr	Pointer to the SAKs Key (and needed Key information) to use in the MACsec's reception secure association
	Active	Boolean to enable/disable the MACsec's reception secure association
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: The request has been accepted `E_NOT_OK`: The request has not been accepted
Description	Request the Ethernet Switch Driver to create a Reception Secure Association in the provided Port. The Short Secure Channel Identifier is included to support XPN configurations. **Tags:** atp.Status=draft	
Available via	EthIf.h	

⌋

8.4.4.14 EthIf_MacSecAddRxSa

[SWS_EthIf_91205] Definition of API function EthIf_MacSecAddRxSa

Status: DRAFT

⌈

Service Name	EthIf_MacSecAddRxSa (draft)	

Syntax	`Std_ReturnType EthIf_MacSecAddRxSa (`
	`  uint8 CtrlIdx,`
	`  uint8 An,`
	`  uint64 LowestPn,`
	`  uint32 Ssci,`
	`  const Mka_SakKeyPtrType* KeysPtr,`
	`  boolean Active`
	`)`	

Service ID [hex]	0x83	
Sync/Async	Asynchronous	
Reentrancy	Reentrant for different CtrlIdx, Non reentrant for the same CtrlIdx	
Parameters (in)	CtrlIdx	Index of the Ethernet controller within the context of the Ethernet Interface
	An	Association Number to use in the MACsec's reception secure association
	LowestPn	Lowest accepted Packet Number in the MACsec's reception secure association
	Ssci	Short Secure Channel Identifiert used in the MACsec's reception secure association
	KeysPtr	Pointer to the SAKs Key (and needed Key information) to use in the MACsec's reception secure association
	Active	Boolean to enable/disable the MACsec's reception secure association
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: The request has been accepted `E_NOT_OK`: The request has not been accepted
Description	Request the Ethernet Inferface (MACsec per SW) or the Ethernet Transceiver Driver to create a Reception Secure Association in the Transceiver. The Short Secure Channel Identifier is included to support XPN configurations. **Tags:** atp.Status=draft	
Available via	EthIf.h	

⌋

8.4.4.15 EthIf_SwitchMacSecUpdateRxSa

[SWS_EthIf_91230] Definition of API function EthIf_SwitchMacSecUpdateRxSa

Status: DRAFT

⌈

Service Name	EthIf_SwitchMacSecUpdateRxSa (draft)	

Syntax	`Std_ReturnType EthIf_SwitchMacSecUpdateRxSa (`
	`  const EthSwt_MgmtInfoType* MgmtInfoPtr,`
	`  uint8 An,`
	`  uint64 LowestPn,`
	`  boolean Active`
	`)`	

Service ID [hex]	0x78	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different MgmtInfoPtr, Non reentrant for the same MgmtInfoPtr	
Parameters (in)	MgmtInfoPtr	Pointer to the management information within the context of an Ethernet Switch Driver. SwitchIdx in context of the EthIf (EthIf Switch/EthIfSwitchIdx), PortIdx in context of EthSwt (EthSwtPort/ EthSwtPortIdx).
	An	Association Number to use in the MACsec's reception secure association
	LowestPn	Lowest accepted Packet Number in the MACsec's reception secure association
	Active	Boolean to enable/disable the MACsec's reception secure association
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: The request has been accepted `E_NOT_OK`: The request has not been accepted
Description	Request the Ethernet Switch Driver to update the Reception Secure Association with the given Packet Number. The Active parameter is included to change the specified AN status. **Tags:** atp.Status=draft	
Available via	EthIf.h	

⌋

8.4.4.16 EthIf_MacSecUpdateRxSa

[SWS_EthIf_91214] Definition of API function EthIf_MacSecUpdateRxSa

Status: DRAFT

⌈

Service Name	EthIf_MacSecUpdateRxSa (draft)

Syntax	`Std_ReturnType EthIf_MacSecUpdateRxSa (`
	`  uint8 CtrlIdx,`
	`  uint8 An,`
	`  uint64 LowestPn,`
	`  boolean Active`
	`)`

Service ID [hex]	0x82

▽

△

Sync/Async	Synchronous	
Reentrancy	Reentrant for different CtrlIdx, Non reentrant for the same CtrlIdx	
Parameters (in)	CtrlIdx	Index of the Ethernet controller within the context of the Ethernet Interface
	An	Association Number to use in the MACsec's reception secure association
	LowestPn	Lowest accepted Packet Number in the MACsec's reception secure association
	Active	Boolean to enable/disable the MACsec's reception secure association
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: The request has been accepted `E_NOT_OK`: The request has not been accepted
Description	Request the Ethernet Inferface (MACsec per SW) or the Ethernet Transceiver Driver to update the Reception Secure Association with the given Packet Number. The Active parameter is included to change the specified AN status. **Tags:** atp.Status=draft	
Available via	EthIf.h	

⌋

8.4.4.17 EthIf_SwitchMacSecDeleteRxSa

[SWS_EthIf_91231] Definition of API function EthIf_SwitchMacSecDeleteRxSa

Status: DRAFT

⌈

Service Name	EthIf_SwitchMacSecDeleteRxSa (draft)	
Syntax	`Std_ReturnType EthIf_SwitchMacSecDeleteRxSa ( const EthSwt_MgmtInfoType* MgmtInfoPtr, uint8 An )`	
Service ID [hex]	0x79	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different MgmtInfoPtr, Non reentrant for the same MgmtInfoPtr	
Parameters (in)	MgmtInfoPtr	Pointer to the management information within the context of an Ethernet Switch Driver. SwitchIdx in context of the EthIf (EthIf Switch/EthIfSwitchIdx), PortIdx in context of EthSwt (EthSwtPort/ EthSwtPortIdx).
	An	Association Number to use in the MACsec's reception secure association
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: The request has been accepted `E_NOT_OK`: The request has not been accepted
Description	Request the Ethernet Switch Driver to remove the Reception Secure Association identified by the provided Association Number. **Tags:** atp.Status=draft	
Available via	EthIf.h	

⌋

8.4.4.18 EthIf_MacSecDeleteRxSa

[SWS_EthIf_91207] Definition of API function EthIf_MacSecDeleteRxSa

Status: DRAFT

⌈

Service Name	EthIf_MacSecDeleteRxSa (draft)	
Syntax	`Std_ReturnType EthIf_MacSecDeleteRxSa ( uint8 CtrlIdx, uint8 An )`	
Service ID [hex]	0x81	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different CtrlIdx, Non reentrant for the same CtrlIdx	
Parameters (in)	CtrlIdx	Index of the Ethernet controller within the context of the Ethernet Interface
	An	Association Number to use in the MACsec's reception secure association
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: The request has been accepted `E_NOT_OK`: The request has not been accepted
Description	Request the Ethernet Inferface (MACsec per SW) or the Ethernet Transceiver Driver to remove the Reception Secure Association identified by the provided Association Number. **Tags:** atp.Status=draft	
Available via	EthIf.h	

⌋

8.4.4.19 EthIf_SwitchMacSecGetTxSaNextPn

[SWS_EthIf_91232] Definition of API function EthIf_SwitchMacSecGetTxSaNextPn

Status: DRAFT

⌈

Service Name	EthIf_SwitchMacSecGetTxSaNextPn (draft)	
Syntax	`Std_ReturnType EthIf_SwitchMacSecGetTxSaNextPn ( const EthSwt_MgmtInfoType* MgmtInfoPtr, uint8 An, uint64* NextPnPtr )`	
Service ID [hex]	0x7a	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different MgmtInfoPtr, Non reentrant for the same MgmtInfoPtr	
Parameters (in)	MgmtInfoPtr	Pointer to the management information within the context of an Ethernet Switch Driver. SwitchIdx in context of the EthIf (EthIf Switch/EthIfSwitchIdx), PortIdx in context of EthSwt (EthSwtPort/ EthSwtPortIdx).

▽

△

	An	Association Number to use in the MACsec's reception secure association
Parameters (inout)	None	
Parameters (out)	NextPnPtr	Pointer to the Next Packet Number read out from the MACsec Entity (SecY)
Return value	Std_ReturnType	`E_OK`: The request has been accepted `E_NOT_OK`: The request has not been accepted
Description	Request the Ethernet Switch Driver to return the Packet Number that is used for the next packet in the given Transmission Secure Association. **Tags:** atp.Status=draft	
Available via	EthIf.h	

⌋

8.4.4.20 EthIf_MacSecGetTxSaNextPn

[SWS_EthIf_91210] Definition of API function EthIf_MacSecGetTxSaNextPn

Status: DRAFT

⌈

Service Name	EthIf_MacSecGetTxSaNextPn (draft)	
Syntax	`Std_ReturnType EthIf_MacSecGetTxSaNextPn ( uint8 CtrlIdx, uint8 An, uint64* NextPnPtr )`	
Service ID [hex]	0x90	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different CtrlIdx, Non reentrant for the same CtrlIdx	
Parameters (in)	CtrlIdx	Index of the Ethernet controller within the context of the Ethernet Interface
	An	Association Number to use in the MACsec's reception secure association
Parameters (inout)	None	
Parameters (out)	NextPnPtr	Pointer to the Next Packet Number read out from the MACsec Entity (SecY)
Return value	Std_ReturnType	`E_OK`: The request has been accepted `E_NOT_OK`: The request has not been accepted
Description	Request the Ethernet Inferface (MACsec per SW) or the Ethernet Transceiver Driver to return the Packet Number that is used for the next packet in the given Transmission Secure Association. **Tags:** atp.Status=draft	
Available via	EthIf.h	

⌋

8.4.4.21 EthIf_SwitchMacSecGetMacSecStatistics

[SWS_EthIf_91233] Definition of API function EthIf_SwitchMacSecGetMacSecStatistics

Status: DRAFT

⌈

Service Name	EthIf_SwitchMacSecGetMacSecStatistics (draft)	
Syntax	`Std_ReturnType EthIf_SwitchMacSecGetMacSecStatistics ( const EthSwt_MgmtInfoType* MgmtInfoPtr, Mka_Stats_SecYType* MacSecStatsPtr )`	
Service ID [hex]	0x7b	
Sync/Async	Asynchronous	
Reentrancy	Reentrant for different MgmtInfoPtr, Non reentrant for the same MgmtInfoPtr	
Parameters (in)	MgmtInfoPtr	Pointer to the management information within the context of an Ethernet Switch Driver. SwitchIdx in context of the EthIf (EthIf Switch/EthIfSwitchIdx), PortIdx in context of EthSwt (EthSwtPort/ EthSwtPortIdx).
Parameters (inout)	None	
Parameters (out)	MacSecStatsPtr	Pointer to a structure including the MACsec statistics of an MKA participant
Return value	Std_ReturnType	`E_OK`: The request has been accepted `E_NOT_OK`: The request has not been accepted
Description	Request the Ethernet switch Driver to provide MACsec statistics. The result is returned through EthIf_SwitchMacSecGetMacSecStatisticsNotification **Tags:** atp.Status=draft	
Available via	EthIf.h	

⌋

8.4.4.22 EthIf_MacSecGetMacSecStatistics

[SWS_EthIf_91209] Definition of API function EthIf_MacSecGetMacSecStatistics

Status: DRAFT

⌈

Service Name	EthIf_MacSecGetMacSecStatistics (draft)	
Syntax	`Std_ReturnType EthIf_MacSecGetMacSecStatistics ( uint8 CtrlIdx, Mka_Stats_SecYType* MacSecStatsPtr )`	
Service ID [hex]	0x89	
Sync/Async	Asynchronous	
Reentrancy	Reentrant for different CtrlIdx, Non reentrant for the same CtrlIdx	
Parameters (in)	CtrlIdx	Index of the Ethernet controller within the context of the Ethernet Interface
Parameters (inout)	None	
Parameters (out)	MacSecStatsPtr	Pointer to a structure including the MACsec statistics of an MKA participant

▽

△

Return value	Std_ReturnType	`E_OK`: The request has been accepted `E_NOT_OK`: The request has not been accepted
Description	Request the Ethernet Inferface (MACsec per SW) or the Ethernet Transceiver Driver to provide MACsec statistics. The result is returned through EthIf_MacSecGetMacSecStatistics Notification **Tags:** atp.Status=draft	
Available via	EthIf.h	

⌋

8.4.4.23 EthIf_SwitchMacSecOperational

[SWS_EthIf_91236] Definition of API function EthIf_SwitchMacSecOperational

Status: DRAFT

⌈

Service Name	EthIf_SwitchMacSecOperational (draft)	
Syntax	`Std_ReturnType EthIf_SwitchMacSecOperational ( const EthSwt_MgmtInfoType* MgmtInfoPtr, boolean MacSecOperational )`	
Service ID [hex]	0x7e	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different MgmtInfoPtr, Non reentrant for the same MgmtInfoPtr	
Parameters (in)	MgmtInfoPtr	Pointer to the management information within the context of an Ethernet Switch Driver. SwitchIdx in context of the EthIf (EthIf Switch/EthIfSwitchIdx), PortIdx in context of EthSwt (EthSwtPort/ EthSwtPortIdx).
	MacSecOperational	Boolean to notify if MACsec is operational
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: The request has been accepted `E_NOT_OK`: The request has not been accepted
Description	To inform EthIf that MacSec is operational and that EthSM can be notified. (Switch case) **Tags:** atp.Status=draft	
Available via	EthIf.h	

⌋

8.4.4.24 EthIf_MacSecOperational

[SWS_EthIf_91212] Definition of API function EthIf_MacSecOperational

Status: DRAFT

⌈

Service Name	EthIf_MacSecOperational (draft)	

Syntax	`Std_ReturnType EthIf_MacSecOperational (`
	`  uint8 CtrlIdx,`
	`  boolean MacSecOperational`
	`)`	

Service ID [hex]	0x1c	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different CtrlIdx, Non reentrant for the same CtrlIdx	
Parameters (in)	CtrlIdx	Index of the Ethernet controller within the context of the Ethernet Interface
	MacSecOperational	Boolean to notify if MACsec is operational
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: The request has been accepted `E_NOT_OK`: The request has not been accepted
Description	To inform EthIf that MacSec is operational and that EthSM can be informed. (Ethernet Inferface (MACsec per SW) or the Ethernet Transceiver Driver) **Tags:** atp.Status=draft	
Available via	EthIf.h	

⌋

8.4.4.25 EthIf_SwitchMacSecSetControlledPortEnabled

[SWS_EthIf_91237] Definition of API function EthIf_SwitchMacSecSetControlled PortEnabled

Status: DRAFT

⌈

Service Name	EthIf_SwitchMacSecSetControlledPortEnabled (draft)	

Syntax	`Std_ReturnType EthIf_SwitchMacSecSetControlledPortEnabled (`
	`  const EthSwt_MgmtInfoType* MgmtInfoPtr,`
	`  boolean ControlledPortEnabled`
	`)`	

Service ID [hex]	0x7f	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different MgmtInfoPtr, Non reentrant for the same MgmtInfoPtr	
Parameters (in)	MgmtInfoPtr	Pointer to the management information within the context of an Ethernet Switch Driver. SwitchIdx in context of the EthIf (EthIf Switch/EthIfSwitchIdx), PortIdx in context of EthSwt (EthSwtPort/ EthSwtPortIdx).
	ControlledPortEnabled	Boolean to activate the Controlled Port of the PAE
Parameters (inout)	None	

▽

△

Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: The request has been accepted `E_NOT_OK`: The request has not been accepted
Description	Requests to set the Controlled Port enabled parameter of a PAE. **Tags:** atp.Status=draft	
Available via	EthIf.h	

⌋

8.4.4.26 EthIf_MacSecSetControlledPortEnabled

[SWS_EthIf_91238] Definition of API function EthIf_MacSecSetControlledPortEnabled

Status: DRAFT

⌈

Service Name	EthIf_MacSecSetControlledPortEnabled (draft)	
Syntax	`Std_ReturnType EthIf_MacSecSetControlledPortEnabled ( uint8 CtrlIdx, boolean ControlledPortEnabled )`	
Service ID [hex]	0x80	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different CtrlIdx, Non reentrant for the same CtrlIdx	
Parameters (in)	CtrlIdx	Index of the Ethernet controller within the context of the Ethernet Interface
	ControlledPortEnabled	Boolean to activate the Controlled Port of the PAE
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: The request has been accepted `E_NOT_OK`: The request has not been accepted
Description	Requests to set the Controlled Port enabled parameter of a PAE. **Tags:** atp.Status=draft	
Available via	EthIf.h	

⌋

8.4.5 Switch driver

8.4.5.1 EthIf_GetSwitchPortMode

[SWS_EthIf_91107] Definition of API function EthIf_GetSwitchPortMode ⌈

Service Name	EthIf_GetSwitchPortMode	
Syntax	`Std_ReturnType EthIf_GetSwitchPortMode ( uint8 SwitchIdx, uint8 SwitchPortIdx, Eth_ModeType* PortModePtr )`	
Service ID [hex]	0x49	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	SwitchIdx	Index of the switch within the context of the Ethernet Switch Driver
	SwitchPortIdx	Index of the port at the addressed switch
Parameters (inout)	None	
Parameters (out)	PortModePtr	ETH_MODE_DOWN: The Ethernet switch port of the given Ethernet switch is disabled ETH_MODE_ACTIVE: The Ethernet switch port of the given Ethernet switch is enabled
Return value	Std_ReturnType	`E_OK`: success `E_NOT_OK`: The mode of the indexed switch port could not be obtained, or the function is called in state `ETHSWT_STATE_UNINIT` or `ETHSWT_STATE_INIT`.
Description	Obtains the mode of the indexed switch port	
Available via	EthIf.h	

⌋

[SWS_EthIf_00415] ⌈The function `EthIf_GetSwitchPortMode` shall forward the call to function `EthSwt_GetSwitchPortMode` of the corresponding Ethernet Switch Driver (EthIfSwitchIdx).⌋

8.4.5.2 EthIf_GetSwitchPortWakeupReason

[SWS_EthIf_91005] Definition of API function EthIf_GetSwitchPortWakeupReason ⌈

Service Name	EthIf_GetSwitchPortWakeupReason	
Syntax	`Std_ReturnType EthIf_GetSwitchPortWakeupReason ( uint8 SwitchIdx, uint8 SwitchPortIdx, EthTrcv_WakeupReasonType* WakeupReasonPtr )`	
Service ID [hex]	0x67	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	SwitchIdx	Index of the Ethernet switch within the context of the Ethernet Interface

▽

△

	SwitchPortIdx	Index of the Ethernet switch port index in the context of the Ethernet switch driver
Parameters (inout)	None	
Parameters (out)	WakeupReasonPtr	Pointer to structure of least recent wakeup event, which was detected by the Ethernet switch port
Return value	Std_ReturnType	E_OK: Ethernet switch port wake up reason request has been accepted. E_NOT_OK: Ethernet switch port wake up reason request has not been accepted.
Description	This function obtains the wake up reasons of the indexed Ethernet switch port by calling EthSwt_GetSwitchPortWakeupReason().	
Available via	EthIf.h	

⌋

[SWS_EthIf_00490]

Upstream requirements: SRS_Eth_00107

⌈The function EthIf_GetSwitchPortWakeupReason shall forward the call to function EthSwt_GetSwitchPortWakeupReason of the corresponding Ethernet Switch Driver (EthIfSwitchIdx).⌋

[SWS_EthIf_00134] ⌈The function EthIf_SetPhysAddr shall forward the call to the respective Ethernet Controller Driver.⌋

8.4.5.3 EthIf_StoreConfiguration

[SWS_EthIf_00214] Definition of API function EthIf_StoreConfiguration ⌈

Service Name	EthIf_StoreConfiguration	
Syntax	`Std_ReturnType EthIf_StoreConfiguration ( uint8 SwitchIdx )`	
Service ID [hex]	0x2c	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	SwitchIdx	Index of the switch within the context of the Ethernet Switch Driver
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: Storage/Reset request accepted E_NOT_OK: Storage/Reset request not accepted
Description	Trigger the storage/reset of the configuration of the learned MAC/Port tables of a switch in a persistent manner and will be used by e.g. CDD.	
Available via	EthIf.h	

⌋

[SWS_EthIf_00215] ⌈The function EthIf_StoreConfiguration shall trigger to store the learned MAC/Port tables of a Ethernet switch.⌋

[SWS_EthIf_00216] ⌈The function shall be pre compile time configurable On/Off by the configuration parameter: `EthIfStoreConfigurationApi`.⌋

8.4.5.4 EthIf_ResetConfiguration

[SWS_EthIf_00219] Definition of API function EthIf_ResetConfiguration ⌈

Service Name	EthIf_ResetConfiguration	
Syntax	`Std_ReturnType EthIf_ResetConfiguration ( uint8 SwitchIdx )`	
Service ID [hex]	0x2d	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	SwitchIdx	Index of the switch within the context of the Ethernet Switch Driver
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: Request to persistently reset the MAC/Port table was accepted `E_NOT_OK`: Request to persistently reset the MAC/Port table was not accepted
Description	The function shall request to reset the configuration of the learned MAC/Port tables of a Ethernet switch in a persistent manner. This could be used by e.g. a CDD. The statically configured entries shall still remain.	
Available via	EthIf.h	

⌋

[SWS_EthIf_00220] ⌈The function `EthIf_ResetConfiguration` shall trigger to reset the learned MAC/Port tables of a Ethernet switch.⌋

[SWS_EthIf_00221] ⌈The function shall be pre compile time configurable On/Off by the configuration parameter: `EthIfResetConfigurationApi`.⌋

8.4.5.5 EthIf_SwitchPortGroupRequestMode

[SWS_EthIf_91102] Definition of API function EthIf_SwitchPortGroupRequestMode ⌈

Service Name	EthIf_SwitchPortGroupRequestMode
Syntax	`Std_ReturnType EthIf_SwitchPortGroupRequestMode ( EthIf_SwitchPortGroupIdxType PortGroupIdx, Eth_ModeType PortMode )`
Service ID [hex]	0x06
Sync/Async	Asynchronous
Reentrancy	Non Reentrant

▽

△

Parameters (in)	PortGroupIdx	Index of the port group within the context of the Ethernet Interface
	PortMode	ETH_MODE_DOWN: disable the Ethernet switch port group ETH_MODE_ACTIVE: enable the Ethernet switch port group ETH_MODE_ACTIVE_WITH_WAKEUP_REQUEST: enable the port group and request for a wake-up on the network
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: success E_NOT_OK: port group mode could not be changed
Description	Request a mode for the EthIfSwtPortGroup. The call shall be forwarded to EthSwt by calling EthSwt_SetSwitchPortMode for all EthSwtPorts referenced by the port group.	
Available via	EthIf.h	

⌋

[SWS_EthIf_00270] ⌈If `EthIf_SwitchPortGroupRequestMode` is called with `ETH_MODE_DOWN` EthIf shall start a timer with EthIfSwitchOffPortTimedelay for all ports of the respective EthIf_SwitchPortGroup if the mode `ETH_MODE_DOWN` has been requested for all EthIfSwitchPortGroups referencing the port and the current mode is `ETH_MODE_ACTIVE`.⌋

[SWS_EthIf_00271] ⌈If the timer to switch off ports (see [SWS_EthIf_00270]) elapses for a port, EthIf shall call the following functions in the given order for the corresponding EthSwtPort:

1. `EthSwt_PortLinkStateRequest` with `ETHTRCV_LINK_STATE_DOWN`
2. `EthSwt_SetSwitchPortMode` with `ETH_MODE_DOWN`

⌋

Note: The implementation has to ensure that EthSwtPorts within EthIfSwitchPortGroups are only disabled if all prior activation request have been withdrawn. This could be realized e.g. by a counter mechanism.

Rationale: Delaying to switch off EthSwtPorts by EthIfSwitchOffPortTimedelay is needed to ensure a simultaneous switch-off of the Ethernet switch port and the Ethernet hardware (PHY or another Ethernet switch) of the connected communication partner:

1. If the Ethernet hardware of the connected communication partner is an PHY, then the EthIfSwitchOffPortTimedelay cover the time which is needed until the PHY of the connected communication partner will be switched off, due to the NM handling.
2. If the Ethernet hardware of the connected communication partner is an Ethernet switch, then both EthSwtPorts should be switched off in the same point in time to avoid link down recognition.

Rationale: Avoid that a EthIfSwitchPortGroup which shall be controlled by EthIfController is incidentally called by BswM

8.4.5.6 EthIf_StartAllPorts

[SWS_EthIf_91103] Definition of API function EthIf_StartAllPorts ⌈

Service Name	EthIf_StartAllPorts	
Syntax	`Std_ReturnType EthIf_StartAllPorts ( void )`	
Service ID [hex]	0x07	
Sync/Async	Asynchronous	
Reentrancy	Reentrant	
Parameters (in)	None	
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: Request was accepted `E_NOT_OK`: Request was rejected
Description	Request to set all configured and affected EthSwtPorts to ETH_MODE_ACTIVE	
Available via	EthIf.h	

⌋

8.4.5.7 EthIf_SetSwitchMgmtInfo

[SWS_EthIf_91003] Definition of API function EthIf_SetSwitchMgmtInfo

Upstream requirements: SRS_Eth_00125

⌈

Service Name	EthIf_SetSwitchMgmtInfo	
Syntax	`Std_ReturnType EthIf_SetSwitchMgmtInfo ( PduIdType PduId, EthSwt_MgmtInfoType* MgmtInfoPtr )`	
Service ID [hex]	0x38	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	PduId	Ethernet Interface PDU ID
	MgmtInfoPtr	Pointer to the management information
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: Management infos successfully set `E_NOT_OK`: Setting of management infos failed
Description	Provides additional management information along to an Ethernet frame that requires special treatment within the Switch. For direct data provision, it has to be called before the transmit request is called. For indirect data provision, it can also be called in the context of the Trigger Transmit API.	
Available via	EthIf.h	

⌋

[SWS_EthIf_00279] ⌈The function shall be pre compile time configurable ON/OFF by the configuration parameter: `EthIfSwitchManagementSupport`.⌋

8.4.5.8 EthIf_GetRxMgmtObject

[SWS_EthIf_91105] Definition of API function EthIf_GetRxMgmtObject ⌈

Service Name	EthIf_GetRxMgmtObject	
Syntax	`Std_ReturnType EthIf_GetRxMgmtObject ( PduIdType PduId, EthSwt_MgmtObjectType **MgmtObjectPtr )`	
Service ID [hex]	0x47	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	PduId	Ethernet Interface PDU ID
Parameters (inout)	None	
Parameters (out)	**MgmtObjectPtr	MgmtObjectPtr Pointer to the management object
Return value	Std_ReturnType	`E_OK`: success `E_NOT_OK`: management object could not be obtained
Description	Request the MgmtObject of the (in this context) unique DataPtr.	
Available via	EthIf.h	

⌋

8.4.5.9 EthIf_GetTxMgmtObject

[SWS_EthIf_91106] Definition of API function EthIf_GetTxMgmtObject ⌈

Service Name	EthIf_GetTxMgmtObject	
Syntax	`Std_ReturnType EthIf_GetTxMgmtObject ( PduIdType PduId, EthSwt_MgmtObjectType **MgmtObjectPtr )`	
Service ID [hex]	0x48	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	PduId	Ethernet Interface PDU ID
Parameters (inout)	None	
Parameters (out)	**MgmtObjectPtr	Pointer to the management object
Return value	Std_ReturnType	`E_OK`: success `E_NOT_OK`: management object could not be obtained
Description	Request the MgmtObject of the (in this context) unique BufIdx.	
Available via	EthIf.h	

⌋

8.4.5.10 EthIf_SwtEthRxProcessFrame

[SWS_EthIf_91243] Definition of API function EthIf_SwtEthRxProcessFrame

Upstream requirements: SRS_Eth_00125

⌈

Service Name	EthIf_SwtEthRxProcessFrame	
Syntax	`Std_ReturnType EthIf_SwtEthRxProcessFrame ( uint8 CtrlIdx, Eth_BufIdxType BufIdx, uint8** DataPtr, uint8** LengthPtr, boolean* IsMgmtFrameOnlyPtr )`	
Service ID [hex]	0xb0	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different CtrlIdxs. Non reentrant for the same CtrlIdx.	
Parameters (in)	CtrlIdx	Ethernet Controller index
	BufIdx	Ethernet Rx Buffer index
Parameters (inout)	DataPtr	IN: Pointer to the position of the EtherType of a common Ethernet frame OUT: Pointer to the position of the EtherType in the management frame
	LengthPtr	IN: Pointer to the length of the frame received OUT: Pointer to the length decreased by the management information length.
Parameters (out)	IsMgmtFrameOnlyPtr	Information about the kind of frame FALSE: Frame is not only for management purpose, but also for normal communication. TRUE: Frame is only for management purpose and must not be processed in common receive process
Return value	Std_ReturnType	`E_OK`: Frame successfully processed `E_NOT_OK`: Frame processing failed
Description	Function inspects the Ethernet frame passed by the data pointer for management information and stores it for later use in the corresponding EthRxFinishedIndication call.	
Available via	EthIf.h	

⌋

8.4.5.11 EthIf_SwtEthRxFinishedIndication

[SWS_EthIf_91244] Definition of API function EthIf_SwtEthRxFinishedIndication

Upstream requirements: SRS_Eth_00125

⌈

Service Name	EthIf_SwtEthRxFinishedIndication
Syntax	`Std_ReturnType EthIf_SwtEthRxFinishedIndication ( uint8 CtrlIdx, Eth_BufIdxType BufIdx )`
Service ID [hex]	0xb1
Sync/Async	Synchronous
Reentrancy	Reentrant for different CtrlIdxs. Non reentrant for the same CtrlIdx.

▽

△

Parameters (in)	CtrlIdx	Ethernet Controller index
	BufIdx	Ethernet Rx Buffer index
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: Frame successfully processed `E_NOT_OK`: Frame processing failed
Description	Indication for a finished receive process for a specific Ethernet frame, which results in providing the management information retrieved during the corresponding EthRxProcessFrame call.	
Available via	EthIf.h	

⌋

8.4.5.12 EthIf_SwtEthTxPrepareFrame

[SWS_EthIf_91245] Definition of API function EthIf_SwtEthTxPrepareFrame

Upstream requirements: SRS_Eth_00125

⌈

Service Name	EthIf_SwtEthTxPrepareFrame	
Syntax	`Std_ReturnType EthIf_SwtEthTxPrepareFrame ( uint8 CtrlIdx, Eth_BufIdxType BufIdx, uint8** DataPtr, uint16* LengthPtr )`	
Service ID [hex]	0xb5	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different CtrlIdxs. Non reentrant for the same CtrlIdx.	
Parameters (in)	CtrlIdx	Ethernet Controller index
	BufIdx	Ethernet Rx Buffer index
Parameters (inout)	DataPtr	IN: Pointer to the position of the EtherType of a common Ethernet frame OUT: Pointer to the position of the EtherType in the management frame
	LengthPtr	IN: Pointer to the length of the buffer without management information OUT: Pointer to the modified length needed for buffer and management information
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: Frame successfully prepared `E_NOT_OK`: Frame preparation failed
Description	Prepares the Ethernet frame for common Ethernet communication (frame shall be handled by switch according to the common address resolution behavior) and stores the information for processing of the corresponding TxFinishedIndication call.	
Available via	EthIf.h	

⌋

8.4.5.13 EthIf_SwtEthTxAdaptBufferLength

[SWS_EthIf_91246] Definition of API function EthIf_SwtEthTxAdaptBufferLength

Upstream requirements: SRS_Eth_00125

⌈

Service Name	EthIf_SwtEthTxAdaptBufferLength	

Syntax	`void EthIf_SwtEthTxAdaptBufferLength (`
	`  uint16* LengthPtr`
	`)`	

Service ID [hex]	0xb2	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different CtrlIdxs. Non reentrant for the same CtrlIdx.	
Parameters (in)	None	
Parameters (inout)	LengthPtr	IN: Pointer to the length of the buffer without management information. OUT: Pointer to the modified length needed for buffer and management information.
Parameters (out)	None	
Return value	None	
Description	Modifies the buffer length to be able to insert management information.	
Available via	EthIf.h	

⌋

8.4.5.14 EthIf_SwtEthTxProcessFrame

[SWS_EthIf_91247] Definition of API function EthIf_SwtEthTxProcessFrame

Upstream requirements: SRS_Eth_00125

⌈

Service Name	EthIf_SwtEthTxProcessFrame	

Syntax	`Std_ReturnType EthIf_SwtEthTxProcessFrame (`
	`  uint8 CtrlIdx,`
	`  Eth_BufIdxType BufIdx,`
	`  uint8** DataPtr,`
	`  uint16* LengthPtr`
	`)`	

Service ID [hex]	0xb4	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different CtrlIdxs. Non reentrant for the same CtrlIdx.	
Parameters (in)	CtrlIdx	Ethernet Controller index
	BufIdx	Ethernet Rx Buffer index
Parameters (inout)	DataPtr	IN: Pointer to the position of the EtherType of a common Ethernet frame OUT: Pointer to the position of the EtherType in the management frame
	LengthPtr	IN: Pointer to the length of the received frame OUT: Pointer to the length decreased by the management information length
Parameters (out)	None	

▽

△

Return value	Std_ReturnType	`E_OK`: Frame successfully processed `E_NOT_OK`: Frame processing failed
Description	Function inserts management information into the Ethernet frame.	
Available via	EthIf.h	

⌋

8.4.5.15 EthIf_SwtEthTxFinishedIndication

[SWS_EthIf_91248] Definition of API function EthIf_SwtEthTxFinishedIndication

Upstream requirements: SRS_Eth_00125

⌈

Service Name	EthIf_SwtEthTxFinishedIndication	
Syntax	`Std_ReturnType EthIf_SwtEthTxFinishedIndication ( uint8 CtrlIdx, Eth_BufIdxType BufIdx )`	
Service ID [hex]	0xb6	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different CtrlIdxs. Non reentrant for the same CtrlIdx.	
Parameters (in)	CtrlIdx	Ethernet Controller index
	BufIdx	Ethernet Rx Buffer index
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: Frame successfully processed `E_NOT_OK`: Frame processing failed
Description	Indication for a finished transmit process for a specific Ethernet frame.	
Available via	EthIf.h	

⌋

8.4.5.16 EthIf_GetSwitchPortSignalQuality

[SWS_EthIf_91058] Definition of API function EthIf_GetSwitchPortSignalQuality

⌈

Service Name	EthIf_GetSwitchPortSignalQuality
Syntax	`Std_ReturnType EthIf_GetSwitchPortSignalQuality ( uint8 SwitchIdx, uint8 SwitchPortIdx, EthIf_SignalQualityResultType* ResultPtr )`
Service ID [hex]	0x1a
Sync/Async	Synchronous

▽

△

Reentrancy	Reentrant for different Ethernet switch indexes and Ethernet Switch port indexes. Non reentrant for the same SwitchPortIdx.	
Parameters (in)	SwitchIdx	Index of the Ethernet switch within the context of the Ethernet Interface
	SwitchPortIdx	Index of the Ethernet switch port within the context of the Ethernet Interface
Parameters (inout)	None	
Parameters (out)	ResultPtr	Pointer to the memory where the signal quality in percent shall be stored.
Return value	Std_ReturnType	E_OK: The signal quality retrieved successfully E_NOT_OK: The signal quality not retrieved successfully
Description	Retrieves the signal quality of the link of the given Ethernet switch port	
Available via	EthIf.h	

⌋

[SWS_EthIf_00395] ⌈The function EthIf_GetSwitchPortSignalQuality shall forward the call to function EthSwt_GetPortSignalQuality of the corresponding Ethernet Switch Driver (EthIfSwitchIdx).⌋

8.4.5.17 EthIf_SwitchPortGetLinkState

[SWS_EthIf_91109] Definition of API function EthIf_SwitchPortGetLinkState ⌈

Service Name	EthIf_SwitchPortGetLinkState	

Syntax	`Std_ReturnType EthIf_SwitchPortGetLinkState (`
	`  uint8 SwitchIdx,`
	`  uint8 SwitchPortIdx,`
	`  EthTrcv_LinkStateType* LinkStatePtr`
	`)`	

Service ID [hex]	0x4b	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	SwitchIdx	Index of the switch within the context of the Ethernet Switch Driver
	SwitchPortIdx	Index of the port at the addressed switch
Parameters (inout)	None	
Parameters (out)	LinkStatePtr	ETHTRCV_LINK_STATE_DOWN: Switch port is disconnected ETHTRCV_LINK_STATE_ACTIVE: Switch port is connected
Return value	Std_ReturnType	E_OK: success E_NOT_OK: Link state of the indexed switch port could not be obtained, or the function is called in state ETHSWT_STATE_UNINIT or ETHSWT_STATE_INIT.
Description	Obtains the link state of the indexed switch port	
Available via	EthIf.h	

⌋

[SWS_EthIf_00419] ⌈The function EthIf_SwitchPortGetLinkState shall forward the call to function EthSwt_GetLinkState of the corresponding Ethernet Switch Driver (EthIfSwitchIdx).⌋

8.4.5.18 EthIf_SwitchPortGetBaudRate

[SWS_EthIf_91111] Definition of API function EthIf_SwitchPortGetBaudRate ⌈

Service Name	EthIf_SwitchPortGetBaudRate	
Syntax	`Std_ReturnType EthIf_SwitchPortGetBaudRate ( uint8 SwitchIdx, uint8 SwitchPortIdx, EthTrcv_BaudRateType* BaudRatePtr )`	
Service ID [hex]	0x4d	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	SwitchIdx	Index of the switch within the context of the Ethernet Switch Driver
	SwitchPortIdx	Index of the port at the addressed switch
Parameters (inout)	None	
Parameters (out)	BaudRatePtr	ETHTRCV_BAUD_RATE_10MBIT: 10MBit connection ETHTRCV_BAUD_RATE_100MBIT: 100MBit connection ETHTRCV_BAUD_RATE_1000MBIT: 1000MBit connection ETHTRCV_BAUD_RATE_2500MBIT: 2500MBit connection
Return value	Std_ReturnType	`E_OK`: success `E_NOT_OK`: Baud rate of the indexed switch port could not be obtained, or the function is called in state `ETHSWT_STATE_UNINIT` or `ETHSWT_STATE_INIT`.
Description	Obtains the baud rate of the indexed switch port	
Available via	EthIf.h	

⌋

[SWS_EthIf_00423] ⌈The function `EthIf_SwitchPortGetBaudRate` shall forward the call to function `EthSwt_GetBaudRate` of the corresponding Ethernet Switch Driver (EthIfSwitchIdx).⌋

8.4.5.19 EthIf_SwitchPortGetDuplexMode

[SWS_EthIf_91113] Definition of API function EthIf_SwitchPortGetDuplexMode ⌈

Service Name	EthIf_SwitchPortGetDuplexMode	
Syntax	`Std_ReturnType EthIf_SwitchPortGetDuplexMode ( uint8 SwitchIdx, uint8 SwitchPortIdx, EthTrcv_DuplexModeType* DuplexModePtr )`	
Service ID [hex]	0x4f	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	SwitchIdx	Index of the switch within the context of the Ethernet Switch Driver
	SwitchPortIdx	Index of the port at the addressed switch
Parameters (inout)	None	

▽

△

Parameters (out)	DuplexModePtr	ETHTRCV_DUPLEX_MODE_HALF: half duplex connections ETHTRCV_DUPLEXMODE_FULL: full duplex connection
Return value	Std_ReturnType	`E_OK`: success `E_NOT_OK`: duplex mode of the indexed switch port could not be obtained, or the function is called in state `ETHSWT_STATE_UNINIT` or `ETHSWT_STATE_INIT`.
Description	Obtains the duplex mode of the indexed switch port	
Available via	EthIf.h	

⌋

[SWS_EthIf_00428] ⌈The function `EthIf_SwitchPortGetDuplexMode` shall forward the call to function `EthSwt_GetDuplexMode` of the corresponding Ethernet Switch Driver (EthIfSwitchIdx).⌋

8.4.5.20 EthIf_SwitchPortReadTrcvRegister

[SWS_EthIf_91249] Definition of API function EthIf_SwitchPortReadTrcvRegister

Upstream requirements: SRS_Eth_00120

⌈

Service Name	EthIf_SwitchPortReadTrcvRegister	
Syntax	`Std_ReturnType EthIf_SwitchPortReadTrcvRegister ( uint8 SwitchIdx, uint8 SwitchPortIdx, uint8 RegIdx, uint16* RegValPtr )`	
Service ID [hex]	0xa4	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different SwitchIdx, non reentrant for same SwitchIdx.	
Parameters (in)	SwitchIdx	Index of the switch.
	SwitchPortIdx	Index of the port at the addressed switch.
	RegIdx	Index of the register to be read.
Parameters (inout)	None	
Parameters (out)	RegValPtr	Filled with the register content of the indexed register.
Return value	Std_ReturnType	`E_OK`: success `E_NOT_OK`: Content of the transceiver could not be obtained, or the function is called in state ETHSWT_STATE_UNINIT or ETHSWT_STATE_INIT.
Description	Generic API for reading the content of a transceiver register.	
Available via	EthIf.h	

⌋

[SWS_EthIf_00710] Forwarding a call of EthIf_SwitchPortReadTrcvRegister to EthSwt_ReadTrcvRegister ⌈The function `EthIf_SwitchPortReadTrcvRegister` shall forward the call to function `EthSwt_ReadTrcvRegister` of the corresponding Ethernet Switch Driver (EthIfSwitchIdx).⌋

8.4.5.21 EthIf_SwitchPortWriteTrcvRegister

[SWS_EthIf_91250] Definition of API function EthIf_SwitchPortWriteTrcvRegister

Upstream requirements: SRS_Eth_00120

⌈

Service Name	EthIf_SwitchPortWriteTrcvRegister	
Syntax	`Std_ReturnType EthIf_SwitchPortWriteTrcvRegister ( uint8 SwitchIdx, uint8 SwitchPortIdx, uint8 RegIdx, uint16 RegVal )`	
Service ID [hex]	0xa5	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different SwitchIdx, non reentrant for same SwitchIdx.	
Parameters (in)	SwitchIdx	Index of the switch.
	SwitchPortIdx	Index of the port at the addressed switch.
	RegIdx	Index of the register to be written.
	RegVal	Value to be written into the indexed register.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: success `E_NOT_OK`: Content given by RegVal could not be written to the given register (RegIdx) of the transceiver, or the function is called in state ETHSWT_STATE_UNINIT or ETHSWT_STATE_INIT.
Description	Generic API for writing the content of a transceiver register.	
Available via	EthIf.h	

⌋

[SWS_EthIf_00711] Forwarding a call of EthIf_SwitchPortWriteTrcvRegiste to EthSwt_WriteTrcvRegister ⌈The function `EthIf_SwitchPortWriteTrcvRegister` shall forward the call to function `EthSwt_WriteTrcvRegister` of the corresponding Ethernet Switch Driver (EthIfSwitchIdx).⌋

8.4.5.22 EthIf_SwitchPortReadMmd

[SWS_EthIf_91251] Definition of API function EthIf_SwitchPortReadMmd ⌈

Service Name	EthIf_SwitchPortReadMmd
Syntax	`Std_ReturnType EthIf_SwitchPortReadMmd ( uint8 SwitchIdx, uint8 SwitchPortIdx, uint8 Mmd, uint16 RegIdx, uint16* RegValPtr )`
Service ID [hex]	0xa6
Sync/Async	Synchronous

▽

△

Reentrancy	Reentrant for different SwitchIdx, non reentrant for same SwitchIdx	
Parameters (in)	SwitchIdx	Index of the switch.
	SwitchPortIdx	Index of the port at the addressed switch.
	Mmd	MDIO Manageable Device.
	RegIdx	Index of the transceiver register on the SMI.
Parameters (inout)	None	
Parameters (out)	RegValPtr	Filled with the register content of the indexed register.
Return value	Std_ReturnType	`E_OK`: Service accepted `E_NOT_OK`: Service denied
Description	Reads a transceiver register using Clause 45 access if supported by hardware or implements a Clause 45 access using Clause 22 operations.	
Available via	EthIf.h	

⌋

[SWS_EthIf_00712] Forwarding a call of EthIf_SwitchPortReadMmd to EthSwt_ReadMmd ⌈The function `EthIf_SwitchPortReadMmd` shall forward the call to function `EthSwt_ReadMmd` of the corresponding Ethernet Switch Driver (EthIfSwitchIdx).⌋

8.4.5.23 EthIf_SwitchPortWriteMmd

[SWS_EthIf_91252] Definition of API function EthIf_SwitchPortWriteMmd ⌈

Service Name	EthIf_SwitchPortWriteMmd	

Syntax	`Std_ReturnType EthIf_SwitchPortWriteMmd (`
	`  uint8 SwitchIdx,`
	`  uint8 SwitchPortIdx,`
	`  uint8 Mmd,`
	`  uint16 RegIdx,`
	`  uint16 RegVal`
	`)`	

Service ID [hex]	0xa7	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different SwitchIdx, non reentrant for same SwitchIdx	
Parameters (in)	SwitchIdx	Index of the switch.
	SwitchPortIdx	Index of the port at the addressed switch.
	Mmd	MDIO Manageable Device.
	RegIdx	Value to be written into the indexed register.
	RegVal	Value to be written into the indexed register.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: Service accepted `E_NOT_OK`: Service denied
Description	Writes a transceiver register using Clause 45 access if supported by hardware or implements a Clause 45 access using Clause 22 operations.	
Available via	EthIf.h	

⌋

[SWS_EthIf_00713] Forwarding a call of EthIf_SwitchPortWriteMmd to EthSwt_WriteMmd ⌈The function `EthIf_SwitchPortWriteMmd` shall forward the call to function `EthSwt_WriteMmd` of the corresponding Ethernet Switch Driver (EthIfSwitchIdx).⌋

8.4.5.24 EthIf_SwitchPortGetCounterValues

[SWS_EthIf_91115] Definition of API function EthIf_SwitchPortGetCounterValues ⌈

Service Name	EthIf_SwitchPortGetCounterValues	
Syntax	`Std_ReturnType EthIf_SwitchPortGetCounterValues ( uint8 SwitchIdx, uint8 SwitchPortIdx, Eth_CounterType* CounterPtr )`	
Service ID [hex]	0x51	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	SwitchIdx	Index of the switch within the context of the Ethernet Switch Driver
	SwitchPortIdx	Index of the port at the addressed switch
Parameters (inout)	None	
Parameters (out)	CounterPtr	counter values according to IETF RFC 1757, RFC 1643 and RFC 2233.
Return value	Std_ReturnType	`E_OK`: success `E_NOT_OK`: counter values read failure
Description	Reads a list with drop counter values of the corresponding port of the switch. The meaning of these values is described at Eth_CounterType.	
Available via	EthIf.h	

⌋

[SWS_EthIf_00432] ⌈The function `EthIf_SwitchPortGetCounterValues` shall forward the call to function `EthSwt_GetCounterValues` of the corresponding Ethernet Switch Driver (EthIfSwitchIdx).⌋

8.4.5.25 EthIf_SwitchPortGetRxStats

[SWS_EthIf_91116] Definition of API function EthIf_SwitchPortGetRxStats ⌈

Service Name	EthIf_SwitchPortGetRxStats
Syntax	`Std_ReturnType EthIf_SwitchPortGetRxStats ( uint8 SwitchIdx, uint8 SwitchPortIdx, Eth_RxStatsType* RxStatsPtr )`
Service ID [hex]	0x52
Sync/Async	Synchronous

▽

△

Reentrancy	Non Reentrant	
Parameters (in)	SwitchIdx	Index of the switch within the context of the Ethernet Switch Driver
	SwitchPortIdx	Index of the port at the addressed switch
Parameters (inout)	None	
Parameters (out)	RxStatsPtr	List of values according to IETF RFC 2819 (Remote Network Monitoring Management Information Base)
Return value	Std_ReturnType	`E_OK`: success `E_NOT_OK`: drop counter could not be obtained
Description	Returns a list of statistic counters defined with Eth_RxTatsType. The majority of these Counters are derived from the IETF RFC2819.	
Available via	EthIf.h	

⌋

[SWS_EthIf_00434] ⌈The function `EthIf_SwitchPortGetRxStats` shall forward the call to function `EthSwt_GetRxStats` of the corresponding Ethernet Switch Driver (EthIfSwitchIdx).⌋

8.4.5.26 EthIf_SwitchPortGetTxStats

[SWS_EthIf_91117] Definition of API function EthIf_SwitchPortGetTxStats ⌈

Service Name	EthIf_SwitchPortGetTxStats	
Syntax	`Std_ReturnType EthIf_SwitchPortGetTxStats ( uint8 SwitchIdx, uint8 SwitchPortIdx, Eth_TxStatsType* TxStatsPtr )`	
Service ID [hex]	0x53	
Sync/Async	Asynchronous	
Reentrancy	Non Reentrant	
Parameters (in)	SwitchIdx	–
	SwitchPortIdx	Index of the port at the addressed switch
Parameters (inout)	None	
Parameters (out)	TxStatsPtr	List of values to read statistic values for transmission.
Return value	Std_ReturnType	`E_OK`: success `E_NOTOK`: Tx-statistics could not be obtained
Description	List of values to read statistic values for transmission.	
Available via	EthIf.h	

⌋

[SWS_EthIf_00436] ⌈The function `EthIf_SwitchPortGetTxStats` shall forward the call to function `EthSwt_GetTxStats` of the corresponding Ethernet Switch Driver (EthIfSwitchIdx).⌋

8.4.5.27 EthIf_SwitchPortGetTxErrorCounterValues

[SWS_EthIf_91118] Definition of API function EthIf_SwitchPortGetTxErrorCounterValues ⌈

Service Name	EthIf_SwitchPortGetTxErrorCounterValues	
Syntax	`Std_ReturnType EthIf_SwitchPortGetTxErrorCounterValues ( uint8 SwitchIdx, uint8 SwitchPortIdx, Eth_TxErrorCounterValuesType* TxStatsPtr )`	
Service ID [hex]	0x54	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	SwitchIdx	Index of the switch within the context of the Ethernet Switch Drive
	SwitchPortIdx	Index of the port at the addressed switch
Parameters (inout)	None	
Parameters (out)	TxStatsPtr	List of values to read statistic error counter values for transmission.
Return value	Std_ReturnType	`E_OK`: success, `E_NOTOK`: Tx-statistics could not be obtained
Description	List of values to read statistic error counter values for transmission from.	
Available via	EthIf.h	

⌋

[SWS_EthIf_00438] ⌈The function `EthIf_SwitchPortGetTxErrorCounterValues` shall forward the call to function `EthSwt_GetTxErrorCounterValues` of the corresponding Ethernet Switch Driver (EthIfSwitchIdx).⌋

8.4.5.28 EthIf_SwitchPortGetMacLearningMode

[SWS_EthIf_91119] Definition of API function EthIf_SwitchPortGetMacLearningMode ⌈

Service Name	EthIf_SwitchPortGetMacLearningMode	
Syntax	`Std_ReturnType EthIf_SwitchPortGetMacLearningMode ( uint8 SwitchIdx, uint8 SwitchPortIdx, EthSwt_MacLearningType* MacLearningModePtr )`	
Service ID [hex]	0x55	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	SwitchIdx	Index of the switch within the context of the Ethernet Switch Driver
	SwitchPortIdx	Index of the port at the addressed switch
Parameters (inout)	None	
Parameters (out)	MacLearningModePtr	Defines whether MAC addresses shall be learned and if they shall be learned in software or hardware.

▽

△

Return value	Std_ReturnType	E_OK: success E_NOT_OK: configuration could be persistently reset
Description	Returns the MAC learning mode, i.e. 1.) HW learning enabled, 2.) Hardware learning disabled, 3.) Software learning enabled. Note: This feature is hardware dependent, i.e. the switch hardware needs to support the different learning modes	
Available via	EthIf.h	

⌋

[SWS_EthIf_00440] ⌈The function EthIf_SwitchPortGetMacLearningMode shall forward the call to function EthSwt_GetMacLearningMode of the corresponding Ethernet Switch Driver (EthIfSwitchIdx).⌋

8.4.5.29 EthIf_GetSwitchPortIdentifier

[SWS_EthIf_91120] Definition of API function EthIf_GetSwitchPortIdentifier ⌈

Service Name	EthIf_GetSwitchPortIdentifier	
Syntax	`Std_ReturnType EthIf_GetSwitchPortIdentifier ( uint8 SwitchIdx, uint8 SwitchPortIdx, uint32* OrgUniqueIdPtr, uint8* ModelNrPtr, uint8* RevisionNrPtr )`	
Service ID [hex]	0x56	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	SwitchIdx	Index of the switch within the context of the Ethernet Switch Driver
	SwitchPortIdx	Index of the port at the addressed switch
Parameters (inout)	None	
Parameters (out)	OrgUniqueIdPtr	Pointer to the memory where the Organizationally Unique Identifier (OUI) shall be stored.
	ModelNrPtr	Pointer to the memory where the Manufacturer's Model Number shall be stored.
	RevisionNrPtr	Pointer to the memory where the Revision Number shall be stored.
Return value	Std_ReturnType	E_OK: organizationally unique identifier of the Ethernet transceiver could be read. E_NOT_OK: organizationally unique identifier of the Ethernet transceiver could not be obtained (i.e. OUI is not available).
Description	This function retrieves the OUI (24 bit) of the indexed Ethernet switch port.	
Available via	EthIf.h	

⌋

[SWS_EthIf_00442] ⌈The function EthIf_GetSwitchPortIdentifier shall forward the call to function EthSwt_GetPortIdentifier of the corresponding Ethernet Switch Driver (EthIfSwitchIdx).⌋

8.4.5.30 EthIf_GetSwitchIdentifier

[SWS_EthIf_91121] Definition of API function EthIf_GetSwitchIdentifier ⌈

Service Name	EthIf_GetSwitchIdentifier	
Syntax	`Std_ReturnType EthIf_GetSwitchIdentifier ( uint8 SwitchIdx, uint32* OrgUniqueIdPtr )`	
Service ID [hex]	0x57	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	SwitchIdx	Index of the switch within the context of the Ethernet Switch Driver
Parameters (inout)	None	
Parameters (out)	OrgUniqueIdPtr	Pointer to the memory where the Organizationally Unique Identifier shall be stored.
Return value	Std_ReturnType	`E_OK`: organizationally unique identifier of the Ethernet switch could be read. `E_NOT_OK`: organizationally unique identifier of the Ethernet switch could not be read (i.e. no OUI is available for this Ethernet switch)
Description	Obtain the Organizationally Unique Identifier that is given by the IEEE of the indexed Ethernet switch. This function shall provide the OUI of Ethernet switch. The OUI has a size of 24 bit. If a ethernet switch can provide the OUI the 8 most significant bits of the OUI shall be set to 0x00xxxxxx. If a Ethernet switch can not provide the OUI the 8 most significant bits of the OUI shall be set to 0xFFxxxxxx.	
Available via	EthIf.h	

⌋

[SWS_EthIf_00444] ⌈The function `EthIf_GetSwitchIdentifier` shall forward the call to function `EthSwt_GetSwitchIdentifier` of the corresponding Ethernet Switch Driver (EthIfSwitchIdx).⌋

8.4.5.31 EthIf_WritePortMirrorConfiguration

[SWS_EthIf_91122] Definition of API function EthIf_WritePortMirrorConfiguration ⌈

Service Name	EthIf_WritePortMirrorConfiguration	
Syntax	`Std_ReturnType EthIf_WritePortMirrorConfiguration ( uint8 MirroredSwitchIdx, const EthSwt_PortMirrorCfgType* PortMirrorConfigurationPtr )`	
Service ID [hex]	0x58	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	MirroredSwitchIdx	Index of the switch within the context of the Ethernet Switch Driver, where the Ethernet switch port is located, that has to be mirrored
	PortMirrorConfiguration Ptr	–

▽

△

Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: the port mirror configuration for the indexed Ethernet switch port was written. `E_NOT_OK`: the port mirror configuration for the indexed Ethernet switch port was not written. (i.e. indexed ethernet switch is not available) `ETHSWT_PORT_MIRRORING_CONFIGURATION_NOT_SUPPORTED`: port mirroring configuration is not supported by Ethernet switch driver or by the Ethernet switch hardware
Description	Store the given port mirror configuration in a shadow buffer in the Ethernet switch driver for the given MirroredSwitchIdx.	
Available via	EthIf.h	

⌋

[SWS_EthIf_00446] ⌈The function `EthIf_WritePortMirrorConfiguration` shall forward the call to function `EthSwt_WritePortMirrorConfiguration` of the corresponding Ethernet Switch Driver (EthIfSwitchIdx).⌋

8.4.5.32 EthIf_ReadPortMirrorConfiguration

[SWS_EthIf_91123] Definition of API function EthIf_ReadPortMirrorConfiguration
⌈

Service Name	EthIf_ReadPortMirrorConfiguration	
Syntax	`Std_ReturnType EthIf_ReadPortMirrorConfiguration ( uint8 MirroredSwitchIdx, EthSwt_PortMirrorCfgType* PortMirrorConfigurationPtr )`	
Service ID [hex]	0x59	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	MirroredSwitchIdx	Index of the Ethernet switch within the context of the Ethernet Switch Driver, where the Ethernet switch ports are located, that have to be mirrored
Parameters (inout)	None	
Parameters (out)	PortMirrorConfiguration Ptr	Pointer to the memory where the port configuration shall be stored.
Return value	Std_ReturnType	`E_OK`: the port mirror configuration for the indexed Ethernet switch port was red successfully. `E_NOT_OK`: the port mirror configuration for the indexed Ethernet switch was not red successfully. (i.e. indexed Ethernet switch is not available)
Description	Obtain the port mirror configuration of the given Ethernet switch.	
Available via	EthIf.h	

⌋

[SWS_EthIf_00448] ⌈The function `EthIf_ReadPortMirrorConfiguration` shall forward the call to function `EthSwt_ReadPortMirrorConfiguration` of the corresponding Ethernet Switch Driver (EthIfSwitchIdx).⌋

8.4.5.33 EthIf_DeletePortMirrorConfiguration

[SWS_EthIf_91124] Definition of API function EthIf_DeletePortMirrorConfiguration ⌈

Service Name	EthIf_DeletePortMirrorConfiguration	
Syntax	`Std_ReturnType EthIf_DeletePortMirrorConfiguration ( uint8 MirroredSwitchIdx )`	
Service ID [hex]	0x5a	
Sync/Async	Synchronous	
Reentrancy	Reentrant Reentrant for different MirroredSwitchIdx. Non reentrant for the same SwitchIdx.	
Parameters (in)	MirroredSwitchIdx	Index of the switch within the context of the Ethernet Switch Driver.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: Port mirror configuration was deleted successfully `E_NOT_OK`: Port mirror configuration was not deleted successfully. (e.g. the port mirroring is enabled)
Description	Delete the stored port mirror configuration of the given MirroredSwitchIdx. If no port mirror configuration was found for the given MirroredSwitchIdx, the return value shall be E_OK.	
Available via	EthIf.h	

⌋

[SWS_EthIf_00450] ⌈The function `EthIf_DeletePortMirrorConfiguration` shall forward the call to function `EthSwt_DeletePortMirrorConfiguration` of the corresponding Ethernet Switch Driver (EthIfSwitchIdx).⌋

8.4.5.34 EthIf_GetPortMirrorState

[SWS_EthIf_91125] Definition of API function EthIf_GetPortMirrorState ⌈

Service Name	EthIf_GetPortMirrorState	
Syntax	`Std_ReturnType EthIf_GetPortMirrorState ( uint8 SwitchIdx, uint8 PortIdx, EthSwt_PortMirrorStateType* PortMirrorStatePtr )`	
Service ID [hex]	0x5b	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	SwitchIdx	Index of the switch within the context of the Ethernet Switch Driver
	PortIdx	Index of the port at the addressed switch
Parameters (inout)	None	
Parameters (out)	PortMirrorStatePtr	Pointer to the memory where the port mirroring state (either PORT_MIRRORING_ENABLED or PORT_MIRRORING_ DISABLED)of the given Ethernet switch port shall be stored.

▽

△

Return value	Std_ReturnType	E_OK: the port mirroring state for the indexed Ethernet switch port returned successfully. E_NOT_OK: the port mirror configuration for the indexed Ethernet switch returned not successfully. (i.e. indexed ethernet switch port is not available)
Description	Obtain the current status of the port mirroring for the indexed Ethernet switch port	
Available via	EthIf.h	

⌋

[SWS_EthIf_00452] ⌈The function EthIf_GetPortMirrorState shall forward the call to function EthSwt_GetPortMirrorState of the corresponding Ethernet Switch Driver (EthIfSwitchIdx).⌋

8.4.5.35 EthIf_SetPortMirrorState

[SWS_EthIf_91126] Definition of API function EthIf_SetPortMirrorState ⌈

Service Name	EthIf_SetPortMirrorState	

Syntax	Std_ReturnType EthIf_SetPortMirrorState (
	  uint8 MirroredSwitchIdx,
	  uint8 PortIdx,
	  EthSwt_PortMirrorStateType PortMirrorState
	)	

Service ID [hex]	0x5c	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	MirroredSwitchIdx	Index of the Ethernet switch within the context of the Ethernet Switch Driver, where the port mirroring configuration is located that has to be enabled and disabled, repectively.
	PortIdx	Index of the port at the addressed switch
	PortMirrorState	Contain the requested port mirroring state either PORT_MIRRORING_ENABLED or PORT_MIRRORING_DISABLED
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	Std_ReturnType E_OK: the requested port mirroring state for the indexed Ethernet switch port was set successfully. E_NOT_OK: the requested port mirroring state for the indexed Ethernet switch was not set successfully. (i.e. indexed Ethernet switch is not available, no port mirrior configuration is available)
Description	Request to set the given port mirroring state of the port mirror configuration for the given Ethernet switch.	
Available via	EthIf.h	

⌋

[SWS_EthIf_00454] ⌈The function EthIf_SetPortMirrorState shall forward the call to function EthSwt_SetPortMirrorState of the corresponding Ethernet Switch Driver (EthIfSwitchIdx).⌋

8.4.5.36 EthIf_SetPortTestMode

[SWS_EthIf_91127] Definition of API function EthIf_SetPortTestMode ⌈

Service Name	EthIf_SetPortTestMode	
Syntax	`Std_ReturnType EthIf_SetPortTestMode ( uint8 SwitchIdx, uint8 PortIdx, EthTrcv_PhyTestModeType Mode )`	
Service ID [hex]	0x5d	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	SwitchIdx	Index of the switch within the context of the Ethernet Switch Driver
	PortIdx	Index of the port at the addressed switch
	Mode	Test mode to be activated
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: the port test mode for the indexed Ethernet switch port was set successfully. `E_NOT_OK`: the port test mode for the indexed Ethernet switch was not set successfully. (i.e. indexed Ethernet switch port is not available)
Description	Activates a given test mode of the indexed Ethernet switch port.	
Available via	EthIf.h	

⌋

[SWS_EthIf_00456] ⌈The function `EthIf_SetPortTestMode` shall forward the call to function `EthSwt_SetPortTestMode` of the corresponding Ethernet Switch Driver (EthIfSwitchIdx).⌋

8.4.5.37 EthIf_SetPortLoopbackMode

[SWS_EthIf_91128] Definition of API function EthIf_SetPortLoopbackMode ⌈

Service Name	EthIf_SetPortLoopbackMode	
Syntax	`Std_ReturnType EthIf_SetPortLoopbackMode ( uint8 SwitchIdx, uint8 PortIdx, EthTrcv_PhyLoopbackModeType Mode )`	
Service ID [hex]	0x5e	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	SwitchIdx	Index of the switch within the context of the Ethernet Switch Driver
	PortIdx	Index of the port at the addressed switch
	Mode	Loop-back mode to be activated
Parameters (inout)	None	
Parameters (out)	None	

▽

△

Return value	Std_ReturnType	E_OK: the port mirroring loop-back back mode for the indexed Ethernet switch port was activated successfully. E_NOT_OK: the port mirroring loop-back back mode for the indexed Ethernet switch port was not activated successfully. (i.e. indexed Ethernet switch port is not available)
Description	Activates a given test loop-back mode of the indexed Ethernet switch port.	
Available via	EthIf.h	

⌋

[SWS_EthIf_00458] ⌈The function EthIf_SetPortLoopbackMode shall forward the call to function EthSwt_SetPortLoopbackMode of the corresponding Ethernet Switch Driver (EthIfSwitchIdx).⌋

8.4.5.38 EthIf_SetPortTxMode

[SWS_EthIf_91129] Definition of API function EthIf_SetPortTxMode ⌈

Service Name	EthIf_SetPortTxMode	
Syntax	`Std_ReturnType EthIf_SetPortTxMode ( uint8 SwitchIdx, uint8 PortIdx, EthTrcv_PhyTxModeType Mode )`	
Service ID [hex]	0x5f	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	SwitchIdx	Index of the switch within the context of the Ethernet Switch Driver
	PortIdx	Index of the port at the addressed switch
	Mode	Transmission mode to be activated
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: the port Tx mode for the indexed Ethernet switch port was activated successfully. E_NOT_OK: the port Tx mode for the indexed Ethernet switch port was not activated successfully. (i.e. indexed Ethernet switch port is not available)
Description	Activates a given transmission mode of the indexed Ethernet switch port.	
Available via	EthIf.h	

⌋

[SWS_EthIf_00460] ⌈The function EthIf_SetPortTxMode shall forward the call to function EthSwt_SetPortTxMode of the corresponding Ethernet Switch Driver (EthIfSwitchIdx).⌋

8.4.5.39 EthIf_GetPortCableDiagnosticsResult

[SWS_EthIf_91130] Definition of API function EthIf_GetPortCableDiagnosticsResult ⌈

Service Name	EthIf_GetPortCableDiagnosticsResult	
Syntax	`Std_ReturnType EthIf_GetPortCableDiagnosticsResult ( uint8 SwitchIdx, uint8 PortIdx, EthTrcv_CableDiagResultType* ResultPtr )`	
Service ID [hex]	0x60	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	SwitchIdx	Index of the switch within the context of the Ethernet Switch Driver
	PortIdx	Index of the port at the addressed switch
Parameters (inout)	None	
Parameters (out)	ResultPtr	Pointer to the location where the cable diagnostics result shall be stored
Return value	Std_ReturnType	`E_OK`:the port cable diagnostic result for the indexed Ethernet switch port was obtained successfully. `E_NOT_OK`: the port cable diagnostic result for the indexed Ethernet switch port was not obtained successfully. (i.e. indexed Ethernet switch port is not available)
Description	Retrieves the cable diagnostics result of the indexed Ethernet switch port respectively the referenced Ethernet Transceiver Driver.	
Available via	EthIf.h	

⌋

[SWS_EthIf_00462] ⌈The function `EthIf_GetPortCableDiagnosticsResult` shall forward the call to function `EthSwt_GetPortCableDiagnosticsResult` of the corresponding Ethernet Switch Driver (EthIfSwitchIdx).⌋

8.4.5.40 EthIf_RunPortCableDiagnostic

[SWS_EthIf_91131] Definition of API function EthIf_RunPortCableDiagnostic ⌈

Service Name	EthIf_RunPortCableDiagnostic	
Syntax	`Std_ReturnType EthIf_RunPortCableDiagnostic ( uint8 SwitchIdx, uint8 PortIdx )`	
Service ID [hex]	0x61	
Sync/Async	Synchronous	
Reentrancy	Reentrant Reentrant for different SwitchIdx and PortIdx. Non reentrant for the same SwitchIdx and PortIdx.	
Parameters (in)	SwitchIdx	Index of the switch within the context of the Ethernet Switch Driver.
	PortIdx	Index of the port at the addressed switch.
Parameters (inout)	None	

▽

△

Parameters (out)	None	
Return value	Std_ReturnType	E_OK: The trigger to run the cable diagnostic has been accepted E_NOT_OK: The trigger to run the cable diagnostic has not been accepted
Description	Trigger the cable diagnostics of the given Ethernet Switch port (PortIdx) by calling EthTrcv_RunCableDiagnostic of the referenced Ethernet transceiver.	
Available via	EthIf.h	

⌋

[SWS_EthIf_00464] ⌈If the function EthIf_RunPortCableDiagnostic is called, EthIf shall ensure that the corresponding EthIfController is in mode ETH_MODE_ACTIVE and forward the call to function EthSwt_RunPortCableDiagnostic of the corresponding Ethernet Switch Driver (EthIfSwitchIdx).⌋

8.4.5.41 EthIf_SwitchGetCfgDataRaw

[SWS_EthIf_91133] Definition of API function EthIf_SwitchGetCfgDataRaw ⌈

Service Name	EthIf_SwitchGetCfgDataRaw	
Syntax	`Std_ReturnType EthIf_SwitchGetCfgDataRaw ( uint8 SwitchIdx, uint32 Offset, uint16 Length, uint8* BufferPtr )`	
Service ID [hex]	0x63	
Sync/Async	Asynchronous	
Reentrancy	Non Reentrant	
Parameters (in)	SwitchIdx	Index of the Ethernet switch within the context of the Ethernet Switch Driver
	Offset	Offset of the Ethernet switch memory from where the reading starts
	Length	Length of data in bytes that shall be copied
Parameters (inout)	None	
Parameters (out)	BufferPtr	Pointer to the location where the data shall be copied
Return value	Std_ReturnType	E_OK: the data read was triggered successfully E_NOT_OK: the data read was not triggered successfully (i.e. indexed Ethernet switch is not available)
Description	Retrieves the data in memory of the indexed Ethernet switch in variable length	
Available via	EthIf.h	

⌋

[SWS_EthIf_00468] ⌈The function EthIf_SwitchGetCfgDataRaw shall forward the call to function EthSwt_GetCfgDataRaw of the corresponding Ethernet Switch Driver (EthIfSwitchIdx).⌋

8.4.5.42 EthIf_SwitchGetCfgDataInfo

[SWS_EthIf_91134] Definition of API function EthIf_SwitchGetCfgDataInfo ⌈

Service Name	EthIf_SwitchGetCfgDataInfo	

Syntax	`Std_ReturnType EthIf_SwitchGetCfgDataInfo (`
	`  uint8 SwitchIdx,`
	`  uint32* DataSizePtr,`
	`  uint32* DataAdressPtr`
	`)`	

Service ID [hex]	0x64	
Sync/Async	Asynchronous	
Reentrancy	Reentrant	
Parameters (in)	SwitchIdx	Index of the Ethernet switch within the context of the Ethernet Switch Driver
Parameters (inout)	None	
Parameters (out)	DataSizePtr	Pointer to the location where the total size of the configuration data shall be copied
	DataAdressPtr	Pointer to the location where the start address of the configuration registers shall be copied
Return value	Std_ReturnType	`E_OK`: the data was obtained successfully `E_NOT_OK`: the data was not obtained successfully. (i.e. indexed Ethernet switch is not available)
Description	Retrieves the total size of data and the memory start address of the indexed Ethernet Switch.	
Available via	EthIf.h	

⌋

[SWS_EthIf_00470] ⌈The function `EthIf_SwitchGetCfgDataInfo` shall forward the call to function `EthSwt_GetCfgDataInfo` of the corresponding Ethernet Switch Driver (EthIfSwitchIdx).⌋

8.4.5.43 EthIf_SwitchPortGetMaxQueueBufferFillLevel

[SWS_EthIf_91135] Definition of API function EthIf_SwitchPortGetMaxQueue BufferFillLevel ⌈

Service Name	EthIf_SwitchPortGetMaxQueueBufferFillLevel	

Syntax	`Std_ReturnType EthIf_SwitchPortGetMaxQueueBufferFillLevel (`
	`  uint8 SwitchPortIdx,`
	`  uint8 PortIdx,`
	`  uint8 SwitchPortEgressQueueIdx,`
	`  uint32* SwitchPortEgressMaxQueueBufferFillLevelPtr`
	`)`	

Service ID [hex]	0x65	
Sync/Async	Asynchronous	
Reentrancy	Reentrant Reentrant for different SwitchIdx and PortIdx. Non reentrant for the same SwitchIdx and PortIdx.	
Parameters (in)	SwitchPortIdx	Index of the Ethernet switch within the context of the Ethernet Switch Driver.
	PortIdx	Index of the Ethernet switch egress port at the addressed Ethernet switch.

▽

△

	SwitchPortEgressQueue Idx	Index of the egress queue of the addressed Ethernet switch port
Parameters (inout)	None	
Parameters (out)	SwitchPortEgressMax QueueBufferFillLevelPtr	Pointer to a memory location, where the maximum amount of allocated queue buffer (in bytes) since the last read out shall be stored
Return value	Std_ReturnType	`E_OK`: success `E_NOT_OK`: The maximal queue buffer level could not be obtained
Description	The function retrieves the maximum amount of allocated queue buffer of the indexed Ethernet switch egress port. If the Ethernet switch hardware does not support Ethernet switch port based maximal queue buffer level, the content of SwitchPortEgressMaxQueueBufferFillLevelPtr shall be set to 0xFFFFFFFF. This API may be called by e.g. a CDD.	
Available via	EthIf.h	

⌋

[SWS_EthIf_00472] ⌈The function `EthIf_SwitchPortGetMaxQueueBufferFillLevel` shall forward the call to function `EthSwt_GetMaxQueueBufferFillLevel` of the corresponding Ethernet Switch Driver (EthIfSwitchIdx).⌋

8.4.6 TimeSync

8.4.6.1 EthIf_SwitchEnableTimeStamping

[SWS_EthIf_91007] Definition of API function EthIf_SwitchEnableTimeStamping

Upstream requirements: SRS_Eth_00125

⌈

Service Name	EthIf_SwitchEnableTimeStamping	
Syntax	`Std_ReturnType EthIf_SwitchEnableTimeStamping ( PduIdType PduId, EthSwt_MgmtInfoType* MgmtInfo )`	
Service ID [hex]	0x39	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	PduId	Ethernet Interface PDU ID
Parameters (inout)	None	
Parameters (out)	MgmtInfo	Management information
Return value	Std_ReturnType	`E_OK`: Time stamping on egress successfully enabled `E_NOT_OK`: Enabling of time stamping on egress has been failed
Description	Activates egress time stamping on a dedicated message object, addressed addressed by the PduId which is associated with an Ethernet controller index and an egress queue.	
Available via	EthIf.h	

⌋

[SWS_EthIf_00285] ⌈The function shall be pre compile time configurable ON/OFF by the configuration parameter: `EthIfGlobalTimeSupport`.⌋

8.4.6.2 EthIf_GetCurrentTimeTuple

[SWS_EthIf_91066] Definition of API function EthIf_GetCurrentTimeTuple

Status: DRAFT
Upstream requirements: SRS_Eth_00175

⌈

Service Name	EthIf_GetCurrentTimeTuple (draft)	
Syntax	`Std_ReturnType EthIf_GetCurrentTimeTuple ( uint8 CtrlIdx, uint8 ClkUnitIdx, TimeTupleType* currentTimeTuplePtr )`	
Service ID [hex]	0x95	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	CtrlIdx	Index of Ethernet Controller within the context of the Ethernet Interface which owns the clock unit
	ClkUnitIdx	Index of the Clock Unit within the context of the Ethernet Interface to provide the time tuple
Parameters (inout)	None	
Parameters (out)	currentTimeTuplePtr	Current time tuple with the • value of the free-running clock used for timestamping • value of the adjustable PHC
Return value	Std_ReturnType	`E_OK`: Current time successfully retrieved `E_NOT_OK`: Current time could not be retrieved
Description	Reads the current time of the timestamp clock and the current time of the PHC in an atomic operation. **Tags:** atp.Status=draft	
Available via	EthIf.h	

⌋

[SWS_EthIf_00585]

Status: DRAFT
Upstream requirements: SRS_Eth_00175

⌈The function `EthIf_GetCurrentTimeTuple` shall forward the call to function <EthDrv>_GetCurrentTimeTuple by setting the `CtrlIdx` to the Ethernet controller which is referenced via `EthIfEthCtrlRef` of the corresponding `EthIfPhysController` and `ClkUnitIdx` to Ethernet controller clock unit which is referenced via `EthIfClkUnitRef` of the given `ClkUnitIdx`.⌋

[SWS_EthIf_00605]

Status: DRAFT
Upstream requirements: SRS_BSW_00171

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

[SWS_EthIf_00606]

Status: DRAFT
Upstream requirements: SRS_BSW_00459

⌈The EthIf module shall apply appropriate mechanisms to allow calls of EthIf_GetCurrentTimeTuple API from other partitions than its main function, e.g. by providing an EthIf satellite.⌋

8.4.6.3 EthIf_EnableEgressTimeStamp

[SWS_EthIf_00160] Definition of API function EthIf_EnableEgressTimeStamp ⌈

Service Name	EthIf_EnableEgressTimeStamp	

Syntax	`void EthIf_EnableEgressTimeStamp (`
	`  PduIdType PduId`
	`)`	

Service ID [hex]	0x23	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	PduId	Ethernet Interface PDU ID
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Activates egress time stamping on a dedicated message object. Some HW does store once the egress time stamp marker and some HW needs it always before transmission. There will be no "disable" functionality, due to the fact, that the message type is always "time stamped" by network design.	
Available via	EthIf.h	

⌋

[SWS_EthIf_00164] ⌈The function shall be pre compile time configurable On/Off by the configuration parameter: `EthIfGlobalTimeSupport`.⌋

8.4.6.4 EthIf_GetEgressTimeStamp

[SWS_EthIf_00166] Definition of API function EthIf_GetEgressTimeStamp ⌈

Service Name	EthIf_GetEgressTimeStamp	

Syntax	`Std_ReturnType EthIf_GetEgressTimeStamp (`
	`  PduIdType TxPduId,`
	`  TimeStampQualType* timeQualPtr,`
	`  TimeStampType* timeStampPtr`
	`)`	

Service ID [hex]	0x24	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	TxPduId	Ethernet Interface PDU ID

▽

△

Parameters (inout)	None	
Parameters (out)	timeQualPtr	quality of HW time stamp, e.g. based on current drift
	timeStampPtr	current time stamp
Return value	Std_ReturnType	`E_OK`: success `E_NOT_OK`: failed to read time stamp.
Description	Reads back the egress time stamp on a dedicated message object. It must be called within the TxConfirmation() function.	
Available via	EthIf.h	

⌋

[SWS_EthIf_00170] ⌈The function shall be pre compile time configurable On/Off by the configuration parameter: `EthIfGlobalTimeSupport`.⌋

8.4.6.5 EthIf_GetIngressTimeStamp

[SWS_EthIf_00172] Definition of API function EthIf_GetIngressTimeStamp

Status: OBSOLETE

⌈

Service Name	EthIf_GetIngressTimeStamp (obsolete)	
Syntax	`Std_ReturnType EthIf_GetIngressTimeStamp ( PduIdType PduId, TimeStampQualType* timeQualPtr, TimeStampType* timeStampPtr )`	
Service ID [hex]	0x25	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	PduId	Ethernet Interface PDU ID
Parameters (inout)	None	
Parameters (out)	timeQualPtr	quality of HW time stamp, e.g. based on current drift
	timeStampPtr	current time stamp
Return value	Std_ReturnType	`E_OK`: success `E_NOT_OK`: failed to read time stamp.
Description	Reads back the ingress time stamp on a dedicated message object. It must be called within the RxIndication() function. **Tags:** atp.Status=obsolete	
Available via	EthIf.h	

⌋

[SWS_EthIf_00176]

Status: OBSOLETE

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

8.4.6.6 EthIf_SetPhcTime

[SWS_EthIf_91062] Definition of API function EthIf_SetPhcTime

Status: DRAFT
Upstream requirements: SRS_Eth_00175

⌈

Service Name	EthIf_SetPhcTime (draft)	
Syntax	`Std_ReturnType EthIf_SetPhcTime ( uint8 CtrlIdx, uint8 ClkUnitIdx, const TimeStampType* timeStampPtr )`	
Service ID [hex]	0x96	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	CtrlIdx	Index of Ethernet Controller within the context of the Ethernet Interface which owns the clock unit
	ClkUnitIdx	Index of the Clock Unit within the context of the Ethernet Interface to provide the time tuple
	timeStampPtr	Time value to which the PHC shall be set
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: PHC successfully set `E_NOT_OK`: PHC could not be set
Description	Sets the absolute time of the PHC. **Tags:** atp.Status=draft	
Available via	EthIf.h	

⌋

[SWS_EthIf_00610]

Status: DRAFT
Upstream requirements: SRS_BSW_00171

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

[SWS_EthIf_00586]

Status: DRAFT
Upstream requirements: SRS_Eth_00175

⌈The function `EthIf_SetPhcTime` shall forward the call to function <EthDrv>_SetPhcTime by setting the `CtrlIdx` to the Ethernet controller which is referenced via `EthIfEthCtrlRef` of the corresponding `EthIfPhysController` and `ClkUnitIdx` to Ethernet controller clock unit which is referenced via `EthIfClkUnitRef` of the given `ClkUnitIdx`.⌋

[SWS_EthIf_00611]
Status: DRAFT
Upstream requirements: SRS_BSW_00459

⌈The EthIf module shall apply appropriate mechanisms to allow calls of EthIf_SetPhc Time API from other partitions than its main function, e.g. by providing an EthIf satellite.⌋

8.4.6.7 EthIf_SetPhcCorrection

[SWS_EthIf_91063] Definition of API function EthIf_SetPhcCorrection
Status: DRAFT
Upstream requirements: SRS_Eth_00175

⌈

Service Name	EthIf_SetPhcCorrection (draft)	
Syntax	`Std_ReturnType EthIf_SetPhcCorrection ( uint8 CtrlIdx, uint8 ClkUnitIdx, sint32 rateDeviation, sint32 offset )`	
Service ID [hex]	0x97	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	CtrlIdx	Index of Ethernet Controller within the context of the Ethernet Interface which owns the clock unit
	ClkUnitIdx	Index of the Clock Unit within the context of the Ethernet Interface to provide the time tuple
	rateDeviation	Rate deviation (resolution: 2^{-32}), by which the PHC is requested to be corrected
	offset	Time offset, by which the PHC is requested to be updated. Unit: Nanoseconds.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: PHC successfully set `E_NOT_OK`: PHC could not be set
Description	Sets PHC parameters to adapt rate and offset of the PHC. **Tags:** atp.Status=draft	
Available via	EthIf.h	

⌋

[SWS_EthIf_00623]
Status: DRAFT
Upstream requirements: SRS_BSW_00171

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

[SWS_EthIf_00624]
Status: DRAFT
Upstream requirements: SRS_Eth_00175

⌈The function `EthIf_SetPhcCorrection` shall forward the call to function <EthDrv>_SetPhcTime by setting the `CtrlIdx` to the Ethernet controller which is referenced via `EthIfEthCtrlRef` of the corresponding `EthIfPhysController` and `ClkUnitIdx` to Ethernet controller clock unit which is referenced via `EthIfClkUnitRef` of the given `ClkUnitIdx`.⌋

[SWS_EthIf_00625]
Status: DRAFT
Upstream requirements: SRS_BSW_00459

⌈The EthIf module shall apply appropriate mechanisms to allow calls of EthIf_SetPhc Time API from other partitions than its main function, e.g. by providing an EthIf satellite.⌋

8.4.6.8 EthIf_GetPhcTime

[SWS_EthIf_91064] Definition of API function EthIf_GetPhcTime
Status: DRAFT
Upstream requirements: SRS_Eth_00175

⌈

Service Name	EthIf_GetPhcTime (draft)	
Syntax	`Std_ReturnType EthIf_GetPhcTime ( uint8 CtrlIdx, uint8 ClkUnitIdx, TimeStampQualType timeQualPtr, TimeStampType timeStampPtr )`	
Service ID [hex]	0x98	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	CtrlIdx	Index of Ethernet Controller within the context of the Ethernet Interface which owns the clock unit
	ClkUnitIdx	Index of the Clock Unit within the context of the Ethernet Interface to provide the time tuple
	timeQualPtr	quality of HW time stamp, e.g. based on current drift
	timeStampPtr	current time stamp
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: PHC value successfully retrieved `E_NOT_OK`: PHC value could not be retrieved
Description	Returns the current time value out of the HW registers of the PHC **Tags:** atp.Status=draft	
Available via	EthIf.h	

⌋

[SWS_EthIf_00630]
Status: DRAFT
Upstream requirements: SRS_BSW_00171

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

[SWS_EthIf_00631]
Status: DRAFT
Upstream requirements: SRS_Eth_00175

⌈The function `EthIf_GetPhcTime` shall forward the call to function <EthDrv>_GetPhcTime by setting the `CtrlIdx` to the Ethernet controller which is referenced via `EthIfEthCtrlRef` of the corresponding `EthIfPhysController` and `ClkUnitIdx` to Ethernet controller clock unit which is referenced via `EthIfClkUnitRef` of the given `ClkUnitIdx`.⌋

[SWS_EthIf_00632]
Status: DRAFT
Upstream requirements: SRS_BSW_00459

⌈The EthIf module shall apply appropriate mechanisms to allow calls of EthIf_GetPhc Time API from other partitions than its main function, e.g. by providing an EthIf satellite.⌋

8.4.6.9 EthIf_SetPpsSignalMode

[SWS_EthIf_91065] Definition of API function EthIf_SetPpsSignalMode
Status: DRAFT
Upstream requirements: SRS_Eth_00176

⌈

Service Name	EthIf_SetPpsSignalMode (draft)	
Syntax	`Std_ReturnType EthIf_SetPpsSignalMode ( uint8 CtrlIdx, uint8 ClkUnitIdx, boolean signalMode )`	
Service ID [hex]	0x99	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	CtrlIdx	Index of Ethernet Controller within the context of the Ethernet Interface which owns the clock unit
	ClkUnitIdx	Index of the Clock Unit within the context of the Ethernet Interface to provide the time tuple
	signalMode	TRUE: PPS signal generation is enabled FALSE: PPS signal generation is disabled
Parameters (inout)	None	

▽

△

Parameters (out)	None	
Return value	Std_ReturnType	E_OK: PPS signal generation successfully enabled/disabled E_NOT_OK: Failed to enable/disable PPS signal generation
Description	Enables/disables the generation of a PPS signal **Tags:** atp.Status=draft	
Available via	EthIf.h	

⌋

[SWS_EthIf_00635]

Status: DRAFT
Upstream requirements: SRS_BSW_00171

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

[SWS_EthIf_00636]

Status: DRAFT
Upstream requirements: SRS_Eth_00175

⌈The function EthIf_SetPpsSignalMode shall forward the call to function <EthDrv>_SetPpsSignalMode by setting the CtrlIdx to the Ethernet controller which is referenced via EthIfEthCtrlRef of the corresponding EthIfPhysController and ClkUnitIdx to Ethernet controller clock unit which is referenced via EthIfClkUnitRef of the given ClkUnitIdx.⌋

8.4.7 Transceiver Driver

8.4.7.1 EthIf_CheckWakeup

[SWS_EthIf_00244] Definition of API function EthIf_CheckWakeup ⌈

Service Name	EthIf_CheckWakeup	
Syntax	`Std_ReturnType EthIf_CheckWakeup ( EcuM_WakeupSourceType WakeupSource )`	
Service ID [hex]	0x30	
Sync/Async	Asynchronous	
Reentrancy	Reentrant	
Parameters (in)	WakeupSource	Source device which initiated the wake up event. The source device could either be a Ethernet switch or a Ethernet transceiver
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK when the request to check for a wake-up of the affected Ethernet hardware (e.g. PHY) has been accepted. E_NOT_OK when the request to check for a wake-up of the affected Ethernet hardware is rejected.

▽

△

Description	This API request the affected Ethernet hardware to check for a signaled wake-up. The used Ethernet hardware could be an Ethernet switch or Ethernet transceiver (PHY). This is used e.g. for Ethernet hardware which is compliant to the specification of Open Alliance TC10. This API is called by the integration code. The function could be called in context of the interrupt or on task level.
Available via	EthIf.h

⌋

[SWS_EthIf_00245]

Upstream requirements: SRS_Eth_00106

⌈For all affected Ethernet transceivers (either referenced by EthIfTransceiver or by EthIfSwitchPortGroups) the function `EthIf_CheckWakeup` shall forward the call to function `<EthTrcv>_CheckWakeup` of the respective Ethernet Transceiver Driver. The call shall be forwarded to each Ethernet transceiver only once⌋

Note:[SWS_EthIf_00245] avoids multiple calls if multiple EthIfSwitchPortGroups and/or multiple EthIfControllers reference the same EthIfTransceiver.

[SWS_EthIf_00500]

Upstream requirements: SRS_Eth_00106

⌈For all affected Ethernet switches (referenced by EthIfSwitch) the function `EthIf_CheckWakeup` shall forward the call to function `EthSwt_SwitchCheckWakeup` of the respective Ethernet Switch Driver. The call shall be forwarded to each Ethernet switch only once.⌋

Note:[SWS_EthIf_00500] avoids multiple calls if multiple EthIfControllers reference the same EthIfSwitch.

8.4.7.2 EthIf_GetPhyWakeupReason

[SWS_EthIf_91004] Definition of API function EthIf_GetPhyWakeupReason

Upstream requirements: SRS_Eth_00107

⌈

Service Name	EthIf_GetPhyWakeupReason	
Syntax	`Std_ReturnType EthIf_GetPhyWakeupReason ( uint8 TrcvIdx, EthTrcv_WakeupReasonType* WakeupReasonPtr )`	
Service ID [hex]	0x69	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	TrcvIdx	Index of the transceiver within the context of the Ethernet Interface

▽

△

Parameters (inout)	None	
Parameters (out)	WakeupReasonPtr	Pointer to structure of least recent wakeup event, which was detected by the Ethernet PHY
Return value	Std_ReturnType	`E_OK`: PHY wake up reason request has been accepted. `E_NOT_OK`: PHY wake up reason request has not been accepted.
Description	This function obtains the wake up reasons of the indexed Ethernet Transceiver (PHY) by calling EthTrcv_GetBusWuReason(...)	
Available via	EthIf.h	

⌋

[SWS_EthIf_00486]

Upstream requirements: SRS_Eth_00107

⌈The function `EthIf_GetPhyWakeupReason` shall forward the call to function `EthTrcv_GetBusWuReason` of the corresponding Ethernet Transceiver Driver (TrcvIdx).⌋

8.4.7.3 EthIf_GetTrcvSignalQuality

[SWS_EthIf_91056] Definition of API function EthIf_GetTrcvSignalQuality ⌈

Service Name	EthIf_GetTrcvSignalQuality	
Syntax	`Std_ReturnType EthIf_GetTrcvSignalQuality ( uint8 TrcvIdx, EthIf_SignalQualityResultType* ResultPtr )`	
Service ID [hex]	0x18	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different TrcvIdx. Non reentrant for the same TrcvIdx.	
Parameters (in)	TrcvIdx	Index of the transceiver within the context of the Ethernet Interface
Parameters (inout)	None	
Parameters (out)	ResultPtr	Pointer to the memory where the signal quality in percent shall be stored.
Return value	Std_ReturnType	`E_OK`: The signal quality retrieved successfully `E_NOT_OK`: The signal quality not retrieved successfully
Description	Retrieves the signal quality of the link of the given Ethernet transceiver	
Available via	EthIf.h	

⌋

[SWS_EthIf_00391] ⌈The function `EthIf_GetTrcvSignalQuality` shall forward the call to function `EthTrcv_GetPhySignalQuality` of the corresponding Ethernet Transceiver Driver (TrcvIdx).⌋

8.4.7.4 EthIf_SetPhyTestMode

[SWS_EthIf_91016] Definition of API function EthIf_SetPhyTestMode

Upstream requirements: SRS_Eth_00117

⌈

Service Name	EthIf_SetPhyTestMode	
Syntax	`Std_ReturnType EthIf_SetPhyTestMode ( uint8 TrcvIdx, EthTrcv_PhyTestModeType Mode )`	
Service ID [hex]	0x17	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different TrcvIdx. Non reentrant for the same TrcvIdx.	
Parameters (in)	TrcvIdx	Index of the transceiver within the context of the Ethernet Interface
	Mode	Test mode to be activated
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: The request has been accepted `E_NOT_OK`: The request has not been accepted.
Description	Activates a given test mode.	
Available via	EthIf.h	

⌋

[SWS_EthIf_00324] ⌈The function `EthIf_SetPhyTestMode` shall forward the call to function `EthTrcv_SetPhyTestMode` of the corresponding Ethernet Transceiver Driver (TrcvIdx).⌋

8.4.7.5 EthIf_SetPhyLoopbackMode

[SWS_EthIf_91018] Definition of API function EthIf_SetPhyLoopbackMode

Upstream requirements: SRS_Eth_00117

⌈

Service Name	EthIf_SetPhyLoopbackMode	
Syntax	`Std_ReturnType EthIf_SetPhyLoopbackMode ( uint8 TrcvIdx, EthTrcv_PhyLoopbackModeType Mode )`	
Service ID [hex]	0x12	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different TrcvIdx. Non reentrant for the same TrcvIdx.	
Parameters (in)	TrcvIdx	Index of the transceiver within the context of the Ethernet Interface
	Mode	Loopback mode to be activated
Parameters (inout)	None	

▽

△

Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: The request has been accepted `E_NOT_OK`: The request has not been accepted.
Description	Activates a given loopback mode.	
Available via	EthIf.h	

⌋

[SWS_EthIf_00327] ⌈The function `EthIf_SetPhyLoopbackMode` shall forward the call to function `EthTrcv_SetPhyLoopbackMode` of the corresponding Ethernet Transceiver Driver (TrcvIdx).⌋

8.4.7.6 EthIf_SetPhyTxMode

[SWS_EthIf_91061] Definition of API function EthIf_SetPhyTxMode

Upstream requirements: SRS_Eth_00117

⌈

Service Name	EthIf_SetPhyTxMode	
Syntax	`Std_ReturnType EthIf_SetPhyTxMode ( uint8 TrcvIdx, EthTrcv_PhyTxModeType Mode )`	
Service ID [hex]	0x13	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different TrcvIdx. Non reentrant for the same TrcvIdx.	
Parameters (in)	TrcvIdx	Index of the transceiver within the context of the Ethernet Interface
	Mode	Transmission mode to be activated
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: The request has been accepted `E_NOT_OK`: The request has not been accepted
Description	Activates a given transmission mode.	
Available via	EthIf.h	

⌋

[SWS_EthIf_00388] ⌈The function `EthIf_SetPhyTxMode` shall forward the call to function `EthTrcv_SetPhyTxMode` of the corresponding Ethernet Transceiver Driver (TrcvIdx).⌋

8.4.7.7 EthIf_GetCableDiagnosticsResult

[SWS_EthIf_91014] Definition of API function EthIf_GetCableDiagnosticsResult

Upstream requirements: SRS_Eth_00117

⌈

Service Name	EthIf_GetCableDiagnosticsResult	
Syntax	`Std_ReturnType EthIf_GetCableDiagnosticsResult ( uint8 TrcvIdx, EthTrcv_CableDiagResultType* ResultPtr )`	
Service ID [hex]	0x14	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different TrcvIdx. Non reentrant for the same TrcvIdx.	
Parameters (in)	TrcvIdx	Index of the transceiver within the context of the Ethernet Interface
Parameters (inout)	None	
Parameters (out)	ResultPtr	Pointer to the location where the cable diagnostics result shall be stored
Return value	Std_ReturnType	`E_OK`: The request has been accepted `E_NOT_OK`: The request has not been accepted
Description	Retrieves the cable diagnostics result of a given transceiver.	
Available via	EthIf.h	

⌋

[SWS_EthIf_00330] ⌈The function `EthIf_GetCableDiagnosticsResult` shall forward the call to function `EthTrcv_GetCableDiagnosticsResult` of the corresponding Ethernet Transceiver Driver (TrcvIdx).⌋

8.4.7.8 EthIf_GetPhyIdentifier

[SWS_EthIf_91020] Definition of API function EthIf_GetPhyIdentifier

Upstream requirements: SRS_Eth_00117

⌈

Service Name	EthIf_GetPhyIdentifier	
Syntax	`Std_ReturnType EthIf_GetPhyIdentifier ( uint8 TrcvIdx, uint32* OrgUniqueIdPtr, uint8* ModelNrPtr, uint8* RevisionNrPtr )`	
Service ID [hex]	0x15	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different TrcvIdx. Non reentrant for the same TrcvIdx.	
Parameters (in)	TrcvIdx	Index of the transceiver within the context of the Ethernet Interface
Parameters (inout)	None	

▽

△

Parameters (out)	OrgUniqueIdPtr	Pointer to the memory where the Organizationally Unique Identifier shall be stored.
	ModelNrPtr	Pointer to the memory where the Manufacturer's Model Number shall be stored.
	RevisionNrPtr	Pointer to the memory where the Revision Number shall be stored.
Return value	Std_ReturnType	`E_OK`: The request has been accepted `E_NOT_OK`: The request has not been accepted
Description	Obtains the PHY identifier of the Ethernet Interface according to IEEE 802.3-2015 chapter 22.2.4.3.1 PHY Identifer.	
Available via	EthIf.h	

⌋

[SWS_EthIf_00334] ⌈The function `EthIf_GetPhyIdentifier` shall forward the call to function `EthTrcv_GetPhyIdentifier` of the corresponding Ethernet Transceiver Driver (TrcvIdx).⌋

8.4.7.9 EthIf_GetTransceiverMode

[SWS_EthIf_91108] Definition of API function EthIf_GetTransceiverMode ⌈

Service Name	EthIf_GetTransceiverMode	
Syntax	`Std_ReturnType EthIf_GetTransceiverMode ( uint8 TrcvIdx, Eth_ModeType* TrcvModePtr )`	
Service ID [hex]	0x4a	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	TrcvIdx	Index of the transceiver within the context of the Ethernet Interface.
Parameters (inout)	None	
Parameters (out)	TrcvModePtr	ETH_MODE_DOWN: the transceiver is disabled ETH_MODE_ACTIVE: the transceiver is enable
Return value	Std_ReturnType	`E_OK`: success `E_NOT_OK`: transceiver could not be initialized
Description	Obtains the state of the indexed transceiver	
Available via	EthIf.h	

⌋

[SWS_EthIf_00417] ⌈The function `EthIf_GetTransceiverMode` shall forward the call to function `<EthTrcv>_GetTransceiverMode` of the corresponding Ethernet Transceiver Driver (TrcvIdx).⌋

8.4.7.10 EthIf_SetTransceiverMode

[SWS_EthIf_91257] Definition of API function EthIf_SetTransceiverMode ⌈

Service Name	EthIf_SetTransceiverMode	
Syntax	`Std_ReturnType EthIf_SetTransceiverMode ( uint8 TrcvIdx, Eth_ModeType TrcvMode )`	
Service ID [hex]	0xa8	
Sync/Async	Asynchronous	
Reentrancy	Non Reentrant	
Parameters (in)	TrcvIdx	Index of the transceiver
	TrcvMode	ETH_MODE_DOWN: disable the transceiver ETH_MODE_ACTIVE: enable the transceiver ETH_MODE_ACTIVE_WITH_WAKEUP_REQUEST: enable the transceiver and request to trigger a wake-up on the network, if the used PHY support such a feature. E.g. used for PHYs compliant to OA TC10
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: Service accepted `E_NOT_OK`: Service denied
Description	Enables / disables the indexed transceiver	
Available via	EthIf.h	

⌋

[SWS_EthIf_00714] Forwarding a call of EthIf_SetTransceiverMode to EthTrcv_SetTransceiverMode ⌈The function `EthIf_SetTransceiverMode` shall forward the call to function `<EthTrcv>_SetTransceiverMode` of the corresponding Ethernet Transceiver Driver (TrcvIdx).⌋

8.4.7.11 EthIf_StartAutoNegotiation

[SWS_EthIf_91258] Definition of API function EthIf_StartAutoNegotiation ⌈

Service Name	EthIf_StartAutoNegotiation	
Syntax	`Std_ReturnType EthIf_StartAutoNegotiation ( uint8 TrcvIdx )`	
Service ID [hex]	0xa9	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	TrcvIdx	Index of the transceiver
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: success `E_NOT_OK`: transceiver could not be initialized
Description	Restarts the negotiation of the transmission parameters used by the indexed transceiver	

▽

△

Available via	EthIf.h

⌋

[SWS_EthIf_00715] Forwarding a call of EthIf_StartAutoNegotiation to EthTrcv_StartAutoNegotiation ⌈The function `EthIf_StartAutoNegotiation` shall forward the call to function `<EthTrcv>_StartAutoNegotiation` of the corresponding Ethernet Transceiver Driver (TrcvIdx).⌋

8.4.7.12 EthIf_TransceiverGetLinkState

[SWS_EthIf_91110] Definition of API function EthIf_TransceiverGetLinkState ⌈

Service Name	EthIf_TransceiverGetLinkState	
Syntax	`Std_ReturnType EthIf_TransceiverGetLinkState ( uint8 TrcvIdx, EthTrcv_LinkStateType* LinkStatePtr )`	
Service ID [hex]	0x4c	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	TrcvIdx	Index of the transceiver within the context of the Ethernet Interface.
Parameters (inout)	None	
Parameters (out)	LinkStatePtr	ETHTRCV_LINK_STATE_DOWN: transceiver is disconnected ETHTRCV_LINK_STATE_ACTIVE: transceiver is connected
Return value	Std_ReturnType	`E_OK`: success `E_NOT_OK`: transceiver could not be initialized
Description	Obtains the link state of the indexed transceiver	
Available via	EthIf.h	

⌋

[SWS_EthIf_00421] ⌈The function `EthIf_TransceiverGetLinkState` shall forward the call to function `<EthTrcv>_GetLinkState` of the corresponding Ethernet Transceiver Driver (TrcvIdx).⌋

8.4.7.13 EthIf_TransceiverGetBaudRate

[SWS_EthIf_91112] Definition of API function EthIf_TransceiverGetBaudRate ⌈

Service Name	EthIf_TransceiverGetBaudRate
Syntax	`Std_ReturnType EthIf_TransceiverGetBaudRate ( uint8 TrcvIdx, EthTrcv_BaudRateType* BaudRatePtr )`
Service ID [hex]	0x4e

▽

△

Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	TrcvIdx	Index of the transceiver within the context of the Ethernet Interface.
Parameters (inout)	None	
Parameters (out)	BaudRatePtr	ETHTRCV_BAUD_RATE_10MBIT: 10MBit connection ETHTRCV_BAUD_RATE_100MBIT: 100MBit connection ETHTRCV_BAUD_RATE_1000MBIT: 1000MBit connection ETHTRCV_BAUD_RATE_2500MBIT: 2500MBit connection
Return value	Std_ReturnType	`E_OK`: success `E_NOT_OK`: transceiver could not be initialized
Description	Obtains the baud rate of the indexed transceiver	
Available via	EthIf.h	

⌋

[SWS_EthIf_00426] ⌈The function `EthIf_TransceiverGetBaudRate` shall forward the call to function `EthTrcv_GetBaudRate` of the corresponding Ethernet Transceiver Driver (TrcvIdx).⌋

8.4.7.14 EthIf_TransceiverGetDuplexMode

[SWS_EthIf_91114] Definition of API function EthIf_TransceiverGetDuplexMode ⌈

Service Name	EthIf_TransceiverGetDuplexMode	
Syntax	`Std_ReturnType EthIf_TransceiverGetDuplexMode ( uint8 TrcvIdx, EthTrcv_DuplexModeType* DuplexModePtr )`	
Service ID [hex]	0x50	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	TrcvIdx	Index of the transceiver within the context of the Ethernet Interface.
Parameters (inout)	None	
Parameters (out)	DuplexModePtr	ETHTRCV_DUPLEX_MODE_HALF: half duplex connections ETHTRCV_DUPLEX_MODE_FULL: full duplex connection
Return value	Std_ReturnType	`E_OK`: success `E_NOT_OK`: transceiver could not be initialized
Description	Obtains the duplex mode of the indexed transceiver	
Available via	EthIf.h	

⌋

[SWS_EthIf_00430] ⌈The function `EthIf_TransceiverGetDuplexMode` shall forward the call to function `EthTrcv_GetDuplexMode` of the corresponding Ethernet Transceiver Driver (TrcvIdx).⌋

8.4.7.15 EthIf_RunCableDiagnostic

[SWS_EthIf_91132] Definition of API function EthIf_RunCableDiagnostic ⌈

Service Name	EthIf_RunCableDiagnostic	
Syntax	`Std_ReturnType EthIf_RunCableDiagnostic ( uint8 TrcvIdx )`	
Service ID [hex]	0x62	
Sync/Async	Synchronous	
Reentrancy	Reentrant Reentrant for different TrcvIdx. Non reentrant for the same TrcvIdx.	
Parameters (in)	TrcvIdx	Index of the Ethernet transceiver within the context of the Ethernet Interface.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: The trigger has been accepted. `E_NOT_OK`: The trigger has not been accepted.
Description	Trigger the cable diagnostics for the given Ethernet transceiver.	
Available via	EthIf.h	

⌋

[SWS_EthIf_00466] ⌈If the function `EthIf_RunCableDiagnostic` is called, EthIf shall ensure that the corresponding EthIfController is in mode `ETH_MODE_ACTIVE` and forward the call to function `EthTrcv_RunCableDiagnostic` of the corresponding Ethernet Transceiver Driver (TrcvIdx).⌋

8.4.7.16 EthIf_TransceiverGetMacMethod

[SWS_EthIf_91021] Definition of API function EthIf_TransceiverGetMacMethod

Upstream requirements: SRS_Eth_00117

⌈

Service Name	EthIf_TransceiverGetMacMethod	
Syntax	`Std_ReturnType EthIf_TransceiverGetMacMethod ( uint8* TrcvIdx, EthTrcv_MacMethodType* MacModePtr )`	
Service ID [hex]	0x66	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	TrcvIdx	Index of the transceiver within the context of the Ethernet Interface.
Parameters (inout)	None	
Parameters (out)	MacModePtr	ETHTRCV_MAC_TYPE_CSMA_CD: Carrier-sense multiple access with collicion detection. ETHTRCV_MAC_TYPE_PLCA: Physical layer collision avoidance.
Return value	Std_ReturnType	`E_OK`: success. `E_NOT_OK`: transceiver request has not been accepted.

▽

△

Description	Obtains the media access mode of the transceiver.
Available via	EthIf.h

⌋

[SWS_EthIf_00474]

Upstream requirements: SRS_Eth_00117

⌈The function `EthIf_TransceiverGetMacMethod` shall forward the call to function `EthTrcv_GetMacMethod` of the corresponding Ethernet Transceiver Driver (TrcvIdx).⌋

8.4.8 Wireless(CC2x)

8.4.8.1 EthIf_GetBufWRxParams

[SWS_EthIf_91002] Definition of API function EthIf_GetBufWRxParams ⌈

Service Name	EthIf_GetBufWRxParams	
Syntax	`Std_ReturnType EthIf_GetBufWRxParams ( uint8 CtrlIdx, const WEth_BufWRxParamIdType* RxParamIds, uint32* ParamValues, uint8 NumParams )`	
Service ID [hex]	0x32	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	CtrlIdx	Index of the Ethernet controller within the context of the Ethernet Interface
	RxParamIds	IDs of the Parameters to read
	NumParams	Number of Parameters
Parameters (inout)	None	
Parameters (out)	ParamValues	Values of the Parameters requested
Return value	Std_ReturnType	`E_OK`: success `E_NOT_OK`: failed reading parameters
Description	Read out values related to the receive direction of the transceiver for a received packet. For example, this could be RSSI or Channel belonging to one single packet.	
Available via	EthIf.h	

⌋

[SWS_EthIf_00341] ⌈The function `EthIf_GetBufWRxParams` shall forward the call to function `WEth_GetBufWRxParams` of the respective Wireless Ethernet Controller Driver.⌋

[SWS_EthIf_00342] ⌈The function shall be pre compile time configurable On/Off by the configuration parameter: `EthIfEnableWEthApi`.⌋

Note: The function requires previous reception (`EthIf_RxIndication`).

8.4.8.2 EthIf_GetBufWTxParams

[SWS_EthIf_91054] Definition of API function EthIf_GetBufWTxParams ⌈

Service Name	EthIf_GetBufWTxParams	
Syntax	`Std_ReturnType EthIf_GetBufWTxParams ( uint8 CtrlIdx, const WEth_BufWTxParamIdType* TxParamIds, uint32* ParamValues, uint8 NumParams )`	
Service ID [hex]	0x31	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	CtrlIdx	Index of the Ethernet controller within the context of the Ethernet Interface
	TxParamIds	IDs of the Parameter that are requested
	NumParams	Number of Parameters that are requested
Parameters (inout)	None	
Parameters (out)	ParamValues	Values of the Parameters requested
Return value	Std_ReturnType	`E_OK`: success `E_NOT_OK`: failed reading parameters
Description	Read out values related to the transmit direction of the transceiver for a transmitted packet.	
Available via	EthIf.h	

⌋

[SWS_EthIf_00347] ⌈The function `EthIf_GetBufWTxParams` shall forward the call to function `WEth_GetBufWTxParams` of the respective Wireless Ethernet Controller Driver.⌋

[SWS_EthIf_00348] ⌈The function shall be pre compile time configurable On/Off by the configuration parameter: `EthIfEnableWEthApi`.⌋

Note: The function requires previous transmission (`EthIf_Transmit`).

8.4.8.3 EthIf_SetBufWTxParams

[SWS_EthIf_91017] Definition of API function EthIf_SetBufWTxParams ⌈

Service Name	EthIf_SetBufWTxParams
Syntax	`Std_ReturnType EthIf_SetBufWTxParams ( uint8 CtrlIdx, const WEth_BufWTxParamIdType* TxParamIds, const uint32* ParamValues, uint8 NumParams )`
Service ID [hex]	0x33
Sync/Async	Synchronous
Reentrancy	Non Reentrant

▽

△

Parameters (in)	CtrlIdx	Index of the Ethernet controller within the context of the Ethernet Interface
	TxParamIds	IDs of the Parameter that are provided to the transmit radio
	ParamValues	Values of the Parameters that are provided to the transmit radio
	NumParams	Number of Parameters that are provided to the transmit radio
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: success `E_NOT_OK`: failed setting parameter
Description	Set values related to the transmit direction of the transceiver for a specific buffer (packet to be sent). For example, this can be the desired transmit power or the channel belonging to one single packet.	
Available via	EthIf.h	

⌋

[SWS_EthIf_00353] ⌈The function `EthIf_SetBufWTxParams` shall forward the call to function `WEth_SetBufWTxParams` of the respective Wireless Ethernet Controller Driver.⌋

[SWS_EthIf_00354] ⌈The function shall be pre compile time configurable On/Off by the configuration parameter: `EthIfEnableWEthApi`.⌋

8.4.8.4 EthIf_SetRadioParams

[SWS_EthIf_91026] Definition of API function EthIf_SetRadioParams ⌈

Service Name	EthIf_SetRadioParams	
Syntax	`Std_ReturnType EthIf_SetRadioParams ( uint8 TrcvId, const WEthTrcv_SetRadioParamIdType* ParamIds, const uint32* ParamValue, uint8 NumParams )`	
Service ID [hex]	0x34	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	TrcvId	Index of the transceiver
	ParamIds	IDs of the Parameters to set
	ParamValue	Values of the Parameters to set
	NumParams	Number of Parameters to set
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: success `E_NOT_OK`: failed writing parameters
Description	Set values related to a transceiver's wireless radio. For example, this could be the selection of the radio settings (channel, ...).	
Available via	EthIf.h	

⌋

[SWS_EthIf_00360] ⌈The function EthIf_SetRadioParams shall forward the call to function WEthTrcv_SetRadioParams of the respective Wireless Ethernet Transceiver Driver.⌋

[SWS_EthIf_00361] ⌈The function shall be pre compile time configurable On/Off by the configuration parameter: EthIfEnableWEthApi.⌋

8.4.8.5 EthIf_SetChanRxParams

[SWS_EthIf_91034] Definition of API function EthIf_SetChanRxParams ⌈

Service Name	EthIf_SetChanRxParams	
Syntax	`Std_ReturnType EthIf_SetChanRxParams ( uint8 TrcvId, uint8 RadioId, const WEthTrcv_SetChanRxParamIdType* ParamIds, const uint32* ParamValues, uint8 NumParams )`	
Service ID [hex]	0x35	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	TrcvId	Index of the transceiver
	RadioId	Index of the Transceiver's Radio (including channel)
	ParamIds	IDs of the Parameters to set
	ParamValues	Values of the Parameters to set
	NumParams	Number of Parameters to set
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: success E_NOT_OK: failed writing parameters
Description	Set values related to the receive direction of a transceiver's wireless channel. For example, this could be a channel parameter like the frequency.	
Available via	EthIf.h	

⌋

[SWS_EthIf_00366] ⌈The function EthIf_SetChanRxParams shall forward the call to function WEthTrcv_SetChanRxParams of the respective Wireless Ethernet Transceiver Driver.⌋

[SWS_EthIf_00367] ⌈The function EthIf_SetChanRxParams shall be pre compile time configurable On/Off by the configuration parameter: EthIfEnableWEthApi.⌋

8.4.8.6 EthIf_SetChanTxParams

[SWS_EthIf_91042] Definition of API function EthIf_SetChanTxParams ⌈

Service Name	EthIf_SetChanTxParams	

Syntax	`Std_ReturnType EthIf_SetChanTxParams (`
	`  uint8 TrcvId,`
	`  uint8 RadioId,`
	`  const WEthTrcv_SetChanTxParamIdType* TxParamIds,`
	`  const uint32* ParamValues,`
	`  uint8 NumParams`
	`)`	

Service ID [hex]	0x36	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	TrcvId	Index of the transceiver
	RadioId	Index of the Transceiver's Radio (including channel)
	TxParamIds	IDs of the Parameters to set
	ParamValues	Values of the Parameters to set
	NumParams	Number of Parameters to set
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: success `E_NOT_OK`: failed writing parameters
Description	Set values related to the transmit direction of a transceiver's wireless channel. For example, this could be the bitrate of a channel.	
Available via	EthIf.h	

⌋

[SWS_EthIf_00373] ⌈The function `EthIf_SetChanTxParams` shall forward the call to function `WEthTrcv_SetChanTxParams` of the respective Wireless Ethernet Transceiver Driver.⌋

[SWS_EthIf_00374] ⌈The function shall be pre compile time configurable On/Off by the configuration parameter: `EthIfEnableWEthApi`.⌋

8.4.8.7 EthIf_GetChanRxParams

[SWS_EthIf_91050] Definition of API function EthIf_GetChanRxParams ⌈

Service Name	EthIf_GetChanRxParams

Syntax	`Std_ReturnType EthIf_GetChanRxParams (`
	`  uint8 TrcvId,`
	`  uint8 RadioId,`
	`  const WEthTrcv_GetChanRxParamIdType* ParamIds,`
	`  uint32* ParamValues,`
	`  uint8 NumParams`
	`)`

Service ID [hex]	0x37
Sync/Async	Synchronous

▽

△

Reentrancy	Non Reentrant	
Parameters (in)	TrcvId	Index of the transceiver
	RadioId	Index of the Transceiver's Radio (including channel)
	ParamIds	IDs of the Parameters to read
	NumParams	Number of Parameters to read
Parameters (inout)	None	
Parameters (out)	ParamValues	Values of the requested Parameters
Return value	Std_ReturnType	`E_OK`: success `E_NOT_OK`: failed reading parameters
Description	Read values related to the receive direction of the transceiver. For example, this could be a Channel Busy Ratio (CBR) or the average Channel Idle Time (CIT).	
Available via	EthIf.h	

⌋

[SWS_EthIf_00380] ⌈The function `EthIf_GetChanRxParams` shall forward the call to function `WEthTrcv_GetChanRxParams` of the respective Wireless Ethernet Transceiver Driver.⌋

[SWS_EthIf_00381] ⌈The function shall be pre compile time configurable On/Off by the configuration parameter: `EthIfEnableWEthApi`.⌋

8.4.8.8 EthIf_GetBufCV2xPC5RxParams

[SWS_EthIf_91201] Definition of API function EthIf_GetBufCV2xPC5RxParams ⌈

Service Name	EthIf_GetBufCV2xPC5RxParams	

Syntax	`Std_ReturnType EthIf_GetBufCV2xPC5RxParams (`
	`  PduIdType PduId,`
	`  const CV2x_BufCV2xPC5RxParamIdType* RxParamIds,`
	`  uint16* ParamValues,`
	`  uint8 NumParams`
	`)`	

Service ID [hex]	0x3a	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	PduId	Ethernet Interface PDU ID
	RxParamIds	IDs of the Parameters to read
	NumParams	Number of Parameters
Parameters (inout)	None	
Parameters (out)	ParamValues	Values of the Parameters requested
Return value	Std_ReturnType	`E_OK`: success `E_NOT_OK`: failed reading parameter
Description	Read out values related to the receive direction of the Cellular V2X for a received packet. For example, this could be CBR belonging to one single packet.	
Available via	EthIf.h	

⌋

[SWS_EthIf_00521]

Status: DRAFT

⌈The function `EthIf_GetBufCV2xPC5RxParams` shall forward the call to function `CV2x_GetBufCV2xPC5RxParams` of the respective Cellular V2X Driver.⌋

[SWS_EthIf_00522]

Status: DRAFT

⌈The function `EthIf_GetBufCV2xPC5RxParams` shall be pre compile time configurable On/Off by the configuration parameter: `EthIfEnableCV2xApi`.⌋

Note: The function requires previous transmission (`EthIf_RxIndication`).

8.4.8.9 EthIf_GetBufCV2xPC5TxParams

[SWS_EthIf_91202] Definition of API function EthIf_GetBufCV2xPC5TxParams ⌈

Service Name	EthIf_GetBufCV2xPC5TxParams	
Syntax	`Std_ReturnType EthIf_GetBufCV2xPC5TxParams ( PduIdType PduId, const CV2x_BufCV2xPC5TxParamIdType* TxParamIds, uint16* ParamValues, uint8 NumParams )`	
Service ID [hex]	0x3b	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	PduId	Ethernet Interface PDU ID
	TxParamIds	IDs of the Parameter to get
	NumParams	Number of Parameters
Parameters (inout)	None	
Parameters (out)	ParamValues	Values of the Parameters requested
Return value	Std_ReturnType	`E_OK`: success `E_NOT_OK`: failed reading parameter
Description	Read out values related to the transmit direction of the Cellular V2X for a transmitted packet. For example, this could be transaction ID belonging to one single packet.	
Available via	EthIf.h	

⌋

[SWS_EthIf_00531]

Status: DRAFT

⌈The function `EthIf_GetBufCV2xPC5TxParams` shall forward the call to function `CV2x_GetBufCV2xPC5TxParams` of the respective Cellular V2X Driver.⌋

[SWS_EthIf_00532]

Status: DRAFT

⌈The function `EthIf_GetBufCV2xPC5TxParams` shall be pre compile time configurable On/Off by the configuration parameter: `EthIfEnableCV2xApi`.⌋

Note: The function requires previous transmission (EthIf_Transmit).

8.4.8.10 EthIf_SetBufCV2xPC5TxParams

[SWS_EthIf_91203] Definition of API function EthIf_SetBufCV2xPC5TxParams ⌈

Service Name	EthIf_SetBufCV2xPC5TxParams	
Syntax	`Std_ReturnType EthIf_SetBufCV2xPC5TxParams ( PduIdType PduId, const CV2x_BufCV2xPC5TxParamIdType* TxParamIds, const uint16* ParamValues, uint8 NumParams )`	
Service ID [hex]	0x3c	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	PduId	Ethernet Interface PDU ID
	TxParamIds	IDs of the Parameter to set
	ParamValues	Value of the Parameter to set
	NumParams	Number of Parameters
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: success E_NOT_OK: failed setting parameter
Description	Set values related to the transmit direction of the Cellular V2X for a specific buffer (packet to be sent). For example, this can be the desired ProSe per-packet priority belonging to one single packet.	
Available via	EthIf.h	

⌋

[SWS_EthIf_00541]

Status: DRAFT

⌈The function EthIf_SetBufCV2xPC5TxParams shall forward the call to function CV2x_SetBufCV2xPC5TxParams of the respective Cellular V2X Driver.⌋

[SWS_EthIf_00542]

Status: DRAFT

⌈The function EthIf_SetBufCV2xPC5TxParams shall be pre compile time configurable On/Off by the configuration parameter: EthIfEnableCV2xApi.⌋

8.4.8.11 EthIf_GetChanCV2xPC5TxParams

[SWS_EthIf_91204] Definition of API function EthIf_GetChanCV2xPC5TxParams
⌈

Service Name	EthIf_GetChanCV2xPC5TxParams	
Syntax	`Std_ReturnType EthIf_GetChanCV2xPC5TxParams ( uint8 CtrlId, uint8 ChannelId, const CV2x_GetChanTxParamIdType* ParamIds, uint32* ParamValues, uint8 NumParams )`	
Service ID [hex]	0x3d	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	CtrlId	Index of the controller within the context of the Cellular V2X Driver (Transceiver Id)
	ChannelId	Index of Transceiver's Radio Channel
	ParamIds	IDs of the Parameters to read
	NumParams	Number of parameters to read
Parameters (inout)	None	
Parameters (out)	ParamValues	Value of the requested Parameters
Return value	Std_ReturnType	`E_OK`: success `E_NOT_OK`: failed setting parameter
Description	Read values related to the receive direction of the channel. For example, this could be a Channel Busy Ratio(CBR)	
Available via		

⌋

[SWS_EthIf_00551]

Status: DRAFT

⌈The function `EthIf_GetChanCV2xPC5TxParams` shall forward the call to function `Cv2x_GetChanTxParams` of the respective Cellular V2X Driver.⌋

[SWS_EthIf_00552]

Status: DRAFT

⌈The function `EthIf_GetChanCV2xPC5TxParams` shall be pre compile time configurable On/Off by the configuration parameter: `EthIfEnableCV2xApi`.⌋

8.5 Callback notifications

This is a list of functions provided for other modules.

8.5.1 EthIf_RxIndication

[SWS_EthIf_00085] Definition of API function EthIf_RxIndication

Upstream requirements: SRS_Eth_00169

⌈

Service Name	EthIf_RxIndication	

Syntax	`void EthIf_RxIndication (`
	`  uint8 CtrlIdx,`
	`  Eth_FrameType FrameType,`
	`  boolean IsBroadcast,`
	`  const uint8* PhysAddrPtr,`
	`  const Eth_DataType* DataPtr,`
	`  uint16 DataLen,`
	`  TimeTupleType* IngressTimeTuplePtr,`
	`  Eth_BufIdxType RxHandleId`
	`)`	

Service ID [hex]	0x10	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	CtrlIdx	Index of the physical Ethernet controller within the context of the Ethernet Interface
	FrameType	Frame type of received Ethernet frame
	IsBroadcast	parameter to indicate a broadcast frame
	PhysAddrPtr	Pointer to Physical source address (MAC address in network byte order) of received Ethernet frame
	DataPtr	Pointer to payload of received Ethernet frame.
	DataLen	Length (bytes) of the payload in received frame.
	IngressTimeTuplePtr	Pointer to ingress timestamp provided as time tuple
	RxHandleId	Unique receive handle id provided by the Ethernet Driver, to identify the ingress queue element per physical Ethernet controller
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Receive indication of an Ethernet frame which was received by the indexed controller	
Available via	EthIf.h	

⌋

[SWS_EthIf_00151] ⌈The Ethernet Driver shall indicate broadcast message with the parameter `IsBroadcast` to the Ethernet Interface.⌋

[SWS_EthIf_00145] ⌈If the VLAN is not active the Ethernet Interface shall increment the corresponding measurement data and filter the message⌋

8.5.2 EthIf_TxConfirmation

[SWS_EthIf_00091] Definition of API function EthIf_TxConfirmation ⌈

Service Name	EthIf_TxConfirmation	
Syntax	`void EthIf_TxConfirmation ( uint8 CtrlIdx, Eth_BufIdxType BufIdx, Std_ReturnType Result )`	
Service ID [hex]	0x11	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	CtrlIdx	Index of the physical Ethernet controller within the context of the Ethernet Interface
	BufIdx	Index of the transmitted buffer
	Result	E_OK: The transmission was successful, E_NOT_OK: The transmission failed.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Confirms frame transmission by the indexed controller	
Available via	EthIf.h	

⌋

[SWS_EthIf_00255] ⌈`EthIf_TxConfirmation` shall pass the Result received within `EthIf_TxConfirmation` to the configured upper layer via `<UL>_TxConfirmation`.⌋

8.5.3 EthIf_CtrlModeIndication

[SWS_EthIf_00231] Definition of callback function EthIf_CtrlModeIndication ⌈

Service Name	EthIf_CtrlModeIndication	
Syntax	`void EthIf_CtrlModeIndication ( uint8 CtrlIdx, Eth_ModeType CtrlMode )`	
Service ID [hex]	0x0e	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant for the same CtrlIdx, reentrant for different	
Parameters (in)	CtrlIdx	Index of the physical Ethernet controller within the context of the Ethernet Interface
	CtrlMode	Notified Ethernet controller mode
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Called asynchronously when mode has been read out. Triggered by previous <EthDrv>_Set ControllerMode call. Can directly be called within the trigger functions.	

▽

△

Available via	EthIf.h

⌋

[SWS_EthIf_00252] ⌈The function shall call `EthSM_CtrlModeIndication`.⌋

8.5.4 EthIf_TrcvModeIndication

[SWS_EthIf_00232] Definition of callback function EthIf_TrcvModeIndication ⌈

Service Name	EthIf_TrcvModeIndication	
Syntax	`void EthIf_TrcvModeIndication ( uint8 TrcvIdx, Eth_ModeType TrcvMode )`	
Service ID [hex]	0x0f	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant for the same CtrlIdx, reentrant for different	
Parameters (in)	TrcvIdx	Index of the Ethernet transceiver within the context of the Ethernet Interface
	TrcvMode	Notified Ethernet transceiver mode
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Called asynchronously when a mode change has been read out. If the function is triggered by previous call of EthTrcv_SetTransceiverMode it can directly be called within the trigger function.	
Available via	EthIf.h	

⌋

8.5.5 EthIf_SwitchPortModeIndication

[SWS_EthIf_91055] Definition of API function EthIf_SwitchPortModeIndication ⌈

Service Name	EthIf_SwitchPortModeIndication	
Syntax	`void EthIf_SwitchPortModeIndication ( uint8 SwitchIdx, uint8 SwitchPortIdx, Eth_ModeType PortMode )`	
Service ID [hex]	0x46	
Sync/Async	Asynchronous	
Reentrancy	Non Reentrant	
Parameters (in)	SwitchIdx	Index of the switch within the context of the Ethernet Switch Driver
	SwitchPortIdx	Index of the port at the addressed switch.
	PortMode	Notified Ethernet Switch port mode.
Parameters (inout)	None	

▽

△

Parameters (out)	None
Return value	None
Description	The EthIf shall determine the expected notifications based on the EthSwtPort configuration. In case the EthSwtPort references an EthTrcv the EthIf expects a notification from the EthTrcv via API EthIf_TrcvModeIndication(). Otherwise the EthIf expects a notification from the EthSwt via API EthIf_SwitchPortModeIndication()
Available via	EthIf.h

⌋

8.5.6 EthIf_SleepIndication

[SWS_EthIf_91006] Definition of API function EthIf_SleepIndication

Status: DRAFT
Upstream requirements: SRS_Eth_00156

⌈

Service Name	EthIf_SleepIndication (draft)	
Syntax	`void EthIf_SleepIndication ( uint8 TrcvIdx )`	
Service ID [hex]	0x68	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	TrcvIdx	Index of the Ethernet transceiver within the context of the Ethernet Interface
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	This API is called by the corresponding EthTrcv, if a sleep indication was detected on the network. This could be used e.g. for Ethernet hardware which is compliant to the OA TC10. In this case the Ethernet hardware (PHY) detect an Sleep.Indication which was triggered by a Sleep.Request of the connected link partner. **Tags:** atp.Status=draft	
Available via	EthIf.h	

⌋

[SWS_EthIf_00497]

Status: DRAFT
Upstream requirements: SRS_Eth_00156

⌈The function shall call `EthSM_SleepIndication` with the corresponding EthIfCtrl.⌋

8.5.7 EthIf_StreamStateIndication

[SWS_EthIf_91024] Definition of callback function EthIf_StreamStateIndication

Status: DRAFT
Upstream requirements: FO_RS_Fw_00011

⌈

Service Name	EthIf_StreamStateIndication (draft)	

Syntax	`void EthIf_StreamStateIndication (`
	`  uint8 SwitchIdx,`
	`  uint8 StreamHandleIdxPtr,`
	`  boolean StreamActivityStatus`
	`)`	

Service ID [hex]	0x93	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	SwitchIdx	Index of the switch within the context of the Ethernet Switch Driver
	StreamHandleIdxPtr	Pointer to the StreamHandleIdx for which the current status is returned
	StreamActivityStatus	Activity status of the StreamHandleIdx (True = active, False = inactive)
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	The function is called by the EthSwt driver module once it has successfully set the streams activity status in the Ethernet switch given with SwitchIdx, triggered by a previous call of EthIf_SetStreamState. **Tags:** atp.Status=draft	
Available via	EthIf_Cbk.h	

⌋

8.5.8 EthIf_StreamStatisticsIndication

[SWS_EthIf_91023] Definition of callback function EthIf_StreamStatisticsIndication

Status: DRAFT
Upstream requirements: FO_RS_Fw_00011

⌈

Service Name	EthIf_StreamStatisticsIndication (draft)

Syntax	`void EthIf_StreamStatisticsIndication (`
	`  uint8 SwitchIdx,`
	`  uint8 NumberOfBuckets,`
	`  const Eth_StreamStatisticCounterType* ListOfBucketsPtr`
	`)`

Service ID [hex]	0x94
Sync/Async	Synchronous
Reentrancy	Reentrant

▽

△

Parameters (in)	SwitchIdx	Index of the switch within the context of the Ethernet Switch Driver
	NumberOfBuckets	Number of counting buckets in the switch
	ListOfBucketsPtr	Pointer to the bucket counter values
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	The function is called by the lower layer once it has successfully retrieved the stream statistics (i.e. bucket counter values) from the EthSwt driver given with SwitchIdx. **Tags:** atp.Status=draft	
Available via	EthIf_Cbk.h	

⌋

8.5.9 EthIf_SwitchMacSecGetMacSecStatisticsNotification

[SWS_EthIf_91234] Definition of callback function EthIf_SwitchMacSecGetMacSecStatisticsNotification

Status: DRAFT

⌈

Service Name	EthIf_SwitchMacSecGetMacSecStatisticsNotification (draft)	

Syntax	`void EthIf_SwitchMacSecGetMacSecStatisticsNotification (`
	`  const EthSwt_MgmtInfoType* MgmtInfoPtr`
	`)`	

Service ID [hex]	0x7c	
Sync/Async	Asynchronous	
Reentrancy	Reentrant for different MgmtInfoPtr, Non reentrant for the same MgmtInfoPtr	
Parameters (in)	None	
Parameters (inout)	None	
Parameters (out)	MgmtInfoPtr	Pointer to the management information within the context of an Ethernet Switch Driver. SwitchIdx in context of the EthIf (EthIf Switch/EthIfSwitchIdx), PortIdx in context of EthSwt (EthSwtPort/ EthSwtPortIdx).
Return value	None	
Description	Callback to notify that EthSwt_MacSecGetMacSecStatistics has finished and provide the requested statistics. **Tags:** atp.Status=draft	
Available via	EthIf.h	

⌋

8.5.10 EthIf_MacSecGetMacSecStatisticsNotification

[SWS_EthIf_91235] Definition of callback function EthIf_MacSecGetMacSecStatisticsNotification

Status: DRAFT

⌈

Service Name	EthIf_MacSecGetMacSecStatisticsNotification (draft)	

Syntax	`void EthIf_MacSecGetMacSecStatisticsNotification (`
	`  uint8 CtrlIdx`
	`)`	

Service ID [hex]	0x7d	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different CtrlIdx, Non reentrant for the same CtrlIdx	
Parameters (in)	CtrlIdx	Index of the Ethernet controller within the context of the Ethernet Interface
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Callback to notify that EthTrcv_MacSecGetMacSecStatistics has finished and provide the requested statistics. **Tags:** atp.Status=draft	
Available via	EthIf.h	

⌋

8.5.11 EthIf_SwitchMacSecUpdateSecYNotification

[SWS_EthIf_91217] Definition of callback function EthIf_SwitchMacSecUpdateSecYNotification

Status: DRAFT

⌈

Service Name	EthIf_SwitchMacSecUpdateSecYNotification (draft)	

Syntax	`void EthIf_SwitchMacSecUpdateSecYNotification (`
	`  const EthSwt_MgmtInfoType* MgmtInfoPtr,`
	`  Std_ReturnType Result`
	`)`	

Service ID [hex]	0x6b	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different MgmtInfoPtr, Non reentrant for the same MgmtInfoPtr	
Parameters (in)	MgmtInfoPtr	Pointer to the management information within the context of an Ethernet Switch Driver. SwitchIdx in context of the EthIf (EthIf Switch/EthIfSwitchIdx), PortIdx in context of EthSwt (EthSwtPort/ EthSwtPortIdx).

▽

△

	Result	E_OK: EthSwt_MacSecUpdateSecY has finished and SecY is updated with the provided parameters of EthSwt_MacSecUpdate SecY E_NOT_OK: SecY has not been updated with the provided parameters of EthSwt_MacSecUpdateSecY.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Callback to notify that EthSwt_MacSecUpdateSecY has finished. **Tags:** atp.Status=draft	
Available via	EthIf.h	

⌋

8.5.12 EthIf_MacSecUpdateSecYNotification

[SWS_EthIf_91218] Definition of callback function EthIf_MacSecUpdateSecYNotification

Status: DRAFT

⌈

Service Name	EthIf_MacSecUpdateSecYNotification (draft)	
Syntax	`void EthIf_MacSecUpdateSecYNotification ( uint8 CtrlIdx, Std_ReturnType Result )`	
Service ID [hex]	0x6c	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different CtrlIdx, Non reentrant for the same CtrlIdx	
Parameters (in)	CtrlIdx	Index of the Ethernet controller within the context of the Ethernet Interface
	Result	E_OK: EthTrcv_MacSecUpdateSecY has finished and SecY is updated with the provided parameters of EthTrcv_MacSecUpdate SecY E_NOT_OK: SecY has not been updated with the provided parameters of EthTrcv_MacSecUpdateSecY.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Callback to notify that EthTrcv_MacSecUpdateSecY finished. **Tags:** atp.Status=draft	
Available via	EthIf.h	

⌋

8.5.13 EthIf_SwitchMacSecAddTxSaNotification

[SWS_EthIf_91223] Definition of callback function EthIf_SwitchMacSecAddTxSaNotification

Status: DRAFT

⌈

Service Name	EthIf_SwitchMacSecAddTxSaNotification (draft)	

Syntax	`void EthIf_SwitchMacSecAddTxSaNotification (`
	`  const EthSwt_MgmtInfoType* MgmtInfoPtr,`
	`  Std_ReturnType Result`
	`)`	

Service ID [hex]	0x71	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different MgmtInfoPtr, Non reentrant for the same MgmtInfoPtr	
Parameters (in)	MgmtInfoPtr	Pointer to the management information within the context of an Ethernet Switch Driver. SwitchIdx in context of the EthIf (EthIf Switch/EthIfSwitchIdx), PortIdx in context of EthSwt (EthSwtPort/ EthSwtPortIdx).
	Result	`E_OK`: EthSwt_MacSecAddTxSa has finished and Transmission Secure Association is created `E_NOT_OK`: The Transmission Secure Association is not created through EthSwt_MacSecAddTxSa.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Callback to notify that EthSwt_MacSecAddTxSa has finished. **Tags:** atp.Status=draft	
Available via	EthIf.h	

⌋

8.5.14 EthIf_MacSecAddTxSaNotification

[SWS_EthIf_91224] Definition of callback function EthIf_MacSecAddTxSaNotification

Status: DRAFT

⌈

Service Name	EthIf_MacSecAddTxSaNotification (draft)	

Syntax	`void EthIf_MacSecAddTxSaNotification (`
	`  uint8 CtrlIdx,`
	`  Std_ReturnType Result`
	`)`	

Service ID [hex]	0x72	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different CtrlIdx, Non reentrant for the same CtrlIdx	
Parameters (in)	CtrlIdx	Index of the Ethernet controller within the context of the Ethernet Interface

▽

△

	Result	E_OK: EthTrcv_MacSecAddTxSa has finished and Transmission Secure Association is created E_NOT_OK: The Transmission Secure Association is not created through EthTrcv_MacSecAddTxSa.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Callback to notify that EthTrcv_MacSecAddTxSa has finished. **Tags:** atp.Status=draft	
Available via	EthIf.h	

⌋

8.5.15 EthIf_SwitchMacSecAddRxSaNotification

[SWS_EthIf_91228] Definition of callback function EthIf_SwitchMacSecAddRxSaNotification

Status: DRAFT

⌈

Service Name	EthIf_SwitchMacSecAddRxSaNotification (draft)	
Syntax	`void EthIf_SwitchMacSecAddRxSaNotification ( const EthSwt_MgmtInfoType* MgmtInfoPtr, Std_ReturnType Result )`	
Service ID [hex]	0x76	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different MgmtInfoPtr, Non reentrant for the same MgmtInfoPtr	
Parameters (in)	MgmtInfoPtr	Pointer to the management information within the context of an Ethernet Switch Driver. SwitchIdx in context of the EthIf (EthIf Switch/EthIfSwitchIdx), PortIdx in context of EthSwt (EthSwtPort/ EthSwtPortIdx).
	Result	E_OK: EthSwt_MacSecAddRxSa has finished and Reception Secure Association is created E_NOT_OK: The Reception Secure Association is not created through EthSwt_MacSecAddRxSa.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Callback to notify that EthSwt_MacSecAddRxSa finished. **Tags:** atp.Status=draft	
Available via	EthIf.h	

⌋

8.5.16 EthIf_MacSecAddRxSaNotification

[SWS_EthIf_91229] Definition of callback function EthIf_MacSecAddRxSaNotification

Status: DRAFT

⌈

Service Name	EthIf_MacSecAddRxSaNotification (draft)	

Syntax	`void EthIf_MacSecAddRxSaNotification (`
	`  uint8 CtrlIdx,`
	`  Std_ReturnType Result`
	`)`	

Service ID [hex]	0x77	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different CtrlIdx, Non reentrant for the same CtrlIdx	
Parameters (in)	CtrlIdx	Index of the Ethernet controller within the context of the Ethernet Interface
	Result	`E_OK`: EthTrcv_MacSecAddRxSa has finished and Reception Secure Association is created `E_NOT_OK`: The Reception Secure Association is not created through EthTrcv_MacSecAddRxSa.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Callback to notify that EthTrcv_MacSecAddRxSa has finished **Tags:** atp.Status=draft	
Available via	EthIf.h	

⌋

8.5.17 EthIf_EthMacSecUpdateSecYNotification

[SWS_EthIf_91253] Definition of callback function EthIf_EthMacSecUpdateSecYNotification

Status: DRAFT
Upstream requirements: FO_RS_MACsec_00004

⌈

Service Name	EthIf_EthMacSecUpdateSecYNotification (draft)	

Syntax	`void EthIf_EthMacSecUpdateSecYNotification (`
	`  uint8 PhysCtrlIdx,`
	`  uint8 Result`
	`)`	

Service ID [hex]	0x9e	
Sync/Async	Asynchronous	
Reentrancy	Reentrant Reentrant for different PhysCtrlIdx, Non reentrant for the same PhysCtrlIdx	
Parameters (in)	PhysCtrlIdx	Index of the physical Ethernet controller within the context of Ethernet Interface

▽

△

	Result	E_OK: Ethernet controller updated SecY with the provided parameters in a previous call of Eth_MacSecUpdateSecY E_NOT_OK: Ethernet controller failed to updated SecY with the provided parameters in a previous call of Eth_MacSecUpdateSecY
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Callback to notify that the Ethernet controller has finished attempts to update SecY with provided parameters in a previous call of Eth_MacSecUpdateSecY. **Tags:** atp.Status=draft	
Available via	EthIf.h	

⌋

8.5.18 EthIf_EthMacSecAddRxSaNotification

[SWS_EthIf_91254] Definition of callback function EthIf_EthMacSecAddRxSaNotification

Status: DRAFT
Upstream requirements: FO_RS_MACsec_00004

⌈

Service Name	EthIf_EthMacSecAddRxSaNotification (draft)	

Syntax	`void EthIf_EthMacSecAddRxSaNotification (`
	`  uint8 PhysCtrlIdx,`
	`  uint8 Result`
	`)`	

Service ID [hex]	0x9f	
Sync/Async	Asynchronous	
Reentrancy	Reentrant Reentrant for different PhysCtrlIdx, Non reentrant for the same PhysCtrlIdx	
Parameters (in)	PhysCtrlIdx	Index of the physical Ethernet controller within the context of Ethernet Interface
	Result	E_OK: The Reception Secure Association is created by the Ethernet controller. E_NOT_OK: The Reception Secure Association is not created by the Ethernet. controller.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Callback to notify that the Ethernet controller has finished attempts to create a Reception Secure Association **Tags:** atp.Status=draft	
Available via	EthIf.h	

⌋

8.5.19 EthIf_EthMacSecAddTxSaNotification

[SWS_EthIf_91255] Definition of callback function EthIf_EthMacSecAddTxSaNotification

Status: DRAFT
Upstream requirements: FO_RS_MACsec_00004

⌈

Service Name	EthIf_EthMacSecAddTxSaNotification (draft)	
Syntax	`void EthIf_EthMacSecAddTxSaNotification ( uint8 PhysCtrlIdx, uint8 Result )`	
Service ID [hex]	0xb7	
Sync/Async	Asynchronous	
Reentrancy	Reentrant Reentrant for different PhysCtrlIdx, Non reentrant for the same PhysCtrlIdx	
Parameters (in)	PhysCtrlIdx	Index of the physical Ethernet controller within the context of Ethernet Interface
	Result	`E_OK`: The Transmission Secure Association is created by the Ethernet controller. `E_NOT_OK`: The Transmission Secure Association is not created by the Ethernet controller.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Callback to notify that the Ethernet controller has finished attempts to create a Transmission Secure Association. **Tags:** atp.Status=draft	
Available via	EthIf.h	

⌋

8.5.20 EthIf_EthMacSecGetMacSecStatisticsNotification

[SWS_EthIf_91256] Definition of callback function EthIf_EthMacSecGetMacSecStatisticsNotification

Status: DRAFT
Upstream requirements: FO_RS_MACsec_00004

⌈

Service Name	EthIf_EthMacSecGetMacSecStatisticsNotification (draft)	
Syntax	`Std_ReturnType EthIf_EthMacSecGetMacSecStatisticsNotification ( uint8 PhysCtrlIdx, Mka_Stats_SecYType* MacSecStatsPtr )`	
Service ID [hex]	0xb8	
Sync/Async	Asynchronous	
Reentrancy	Reentrant Reentrant for different PhysCtrlIdx, Non reentrant for the same PhysCtrlIdx	
Parameters (in)	PhysCtrlIdx	Index of the physical Ethernet controller within the context of Ethernet Interface

▽

△

Parameters (inout)	None	
Parameters (out)	MacSecStatsPtr	Pointer to a structure including the MACsec statistics of an MKA participant
Return value	Std_ReturnType	`E_OK`: The request has been accecpted `E_NOT_OK`: The request has been rejected
Description	Callback to get the MACsec statistics of an Ethernet Controller, which was triggered by a previous call of Eth_MacSecGetMacSecStatistics **Tags:** atp.Status=draft	
Available via	EthIf.h	

⌋

8.6 Scheduled functions

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

8.6.1 EthIf_MainFunctionRx

[SWS_EthIf_00097] Definition of scheduled function EthIf_MainFunctionRx ⌈

Service Name	EthIf_MainFunctionRx

Syntax	`void EthIf_MainFunctionRx (`
	`  void`
	`)`

Service ID [hex]	0x20
Description	The function checks for new received frames and issues reception indications in polling mode.
Available via	SchM_EthIf.h

⌋

[SWS_EthIf_00099] ⌈The receive frame check shall be pre compile time configurable On/Off by the configuration parameter: `EthIfEnableRxInterrupt`.⌋

8.6.2 EthIf_MainFunctionRx_<IngressQueueProcessing ShortName>

[SWS_EthIf_91139] Definition of scheduled function EthIf_MainFunction Rx_<IngressQueueProcessing ShortName>

Status: DRAFT
Upstream requirements: SRS_Eth_00170

⌈

Service Name	EthIf_MainFunctionRx_<IngressQueueProcessing ShortName> (draft)
Syntax	`void EthIf_MainFunctionRx_<IngressQueueProcessing ShortName> ( void )`
Service ID [hex]	0x9c
Description	The function checks for new received Ethernet frames at the related Ethernet controller and the related ingress queue referenced via EthIfPhysCtrlRxIngressQueueRef, or at the related Can XL controller and the related ingress FIFO referenced via EthIfCanXLCtrlRxIngressFifoRef. In case of Ethernet controller calling Eth_Receive() with the respective QueueIdx. In case of Can XL controller calling CanXL_Receive() with the respective FifoIdx. **Tags:** atp.Status=draft
Available via	EthIf.h

⌋

8.6.3 EthIf_MainFunctionTx

[SWS_EthIf_00113] Definition of scheduled function EthIf_MainFunctionTx ⌈

Service Name	EthIf_MainFunctionTx
Syntax	`void EthIf_MainFunctionTx ( void )`
Service ID [hex]	0x21
Description	The function issues transmission confirmations in polling mode. It checks also for transceiver state changes.
Available via	SchM_EthIf.h

⌋

[SWS_EthIf_00100] ⌈The transmission confirmation check shall be pre compile time configurable On/Off by the configuration parameter: `EthIfEnableTxInterrupt`.⌋

[SWS_EthIf_00101] ⌈The frequency of polling the transceiver state change shall be configurable by the configuration parameter: `EthIfTrcvLinkStateChgMainReload`.⌋

8.6.4 EthIf_MainFunctionState

[SWS_EthIf_91104] Definition of scheduled function EthIf_MainFunctionState ⌈

Service Name	EthIf_MainFunctionState
Syntax	`void EthIf_MainFunctionState ( void )`
Service ID [hex]	0x05
Description	The function is polling different communication hardware (Ethernet transceiver, Ethernet switch ports) related information, e.g. link state, signal quality.
Available via	EthIf_SchM.h

⌋

[SWS_EthIf_00407] ⌈The function `EthIf_MainFunctionState` shall poll Ethernet communication hardware related information with the period of EthIfMainFunctionStatePeriod.⌋

[SWS_EthIf_00408] ⌈For each Ethernet switch port where a link state `ETHTRCV_LINK_STATE_ACTIVE` is yielded and references an Ethernet Transceiver the function shall poll the signal quality by calling `EthSwt_GetPortSignalQuality`.⌋

[SWS_EthIf_00409] ⌈For each Ethernet transceiver where a link state of `ETHTRCV_LINK_STATE_ACTIVE` is yielded the function shall poll the signal quality by calling `EthTrcv_GetPhySignalQuality`.⌋

[SWS_EthIf_00410] ⌈The obtained signal quality value shall be stored as type of `EthIf_SignalQualityResultType`. The value shall always be stored as `ActualSignalQuality`. If the obtained signal quality is higher than the stored highest signal quality (`HighestSignalQuality`), then HighestSignalQuality shall be updated with the obtained signal quality. If the obtained signal quality is lower than the lowest signal quality (`LowestSignalQuality`), then LowestSignalQuality shall be updated with the obtained signal quality.⌋

[SWS_EthIf_00498] ⌈EthIf shall check its maintained Ethernet hardware (Ethernet switch port, Ethernet transceiver), if the Ethernet hardware has reached the requested mode and requested link state under the following conditions:

- the timer to switch off the EthSwtPort (see EthIfSwitchOffPortTimeDelay) is not running AND
- the timer to keep the EthSwtPort in `ETH_MODE_ACTIVE` (see EthIfPortStartupActiveTime) is not running and the EthSwtPort has not been requested with `ETH_MODE_ACTIVE`

If EthIf detects that the requested mode and / or requested link state has not reached, EthIf shall re-trigger the requested mode and link state, respectively.⌋

Note:

1. This shall ensure to re-trigger a wake-up on the network, if e.g. OA TC10 compliant hardware is used (see [5, OPEN Sleep/Wake-up Specification for Automotive Ethernet]).
2. Additionally, the check shall not try to re-establish a requested mode if the timer to switch off the EthSwtPort (requested via EthIfSwitchOffPortTimeDelay) or the timer to keep the EthSwtPort active (requested via EthIfPortStartupActiveTime) is running. Switching-off of the Ethernet hardware in an Ethernet switched network after EthIfSwitchOffPortTimeDelay expires, lead to a situation that an Ethernet switch port and the connected Ethernet hardware (PHY) of the link partner are not synchronized. Thus, first the connected PHY will be switched off and after EthIfSwitchOffPortTimeDelay the Ethernet switch port. This is acceptable since the network management has already confirmed to go to sleep. For example, if using OA TC10 compliant Ethernet hardware, the ECU which is connected to the Ethernet switch trigger a Sleep.Request on the network and bring the connected Ethernet switch ports and its own Ethernet hardware to sleep mode, due to the specified OA TC10 synchronized shutdown of the Ethernet hardware. Thus, the ECU that maintain the Ethernet switch may detect a link down on the affected Ethernet switch port, which should be ignored by the EthIf, if the switch-off of the Ethernet switch port was already triggered but not forwarded to the Ethernet switch.

[SWS_EthIf_00499]

Status: DRAFT
Upstream requirements: SRS_Eth_00156

⌈For EthIfTransceiver where the referenced EthTrcv is acting as a passive communication slave (EthTrcvActAsSlavePassiveEnabled set to TRUE), EthIf shall check for unexpected link down. If an unexpected link down (link state is requested with `ETHTRCV_LINK_STATE_ACTIVE`, but current link state is `ETHTRCV_LINK_STATE_DOWN`) lasts as long as specified in EthIfQualifiedUnexptecedLinkDownTime, EthIf shall trigger to release the affected communication channel by calling `EthSM_SleepIndication`. If an unexpected link down was detected, the EthSM shall immediatedly be indicated via `EthSM_TrcvLinkStateChg` without considering EthIfQualifiedUnexpectedLinkDownTime.⌋

Note: [SWS_EthIf_00499] should grant that a communication channel that act as an passive communication channel will shutdown even though the communiation master could not transmit a sleep over the network (e.g. hardware failure, unexpected shutdown of the ECU that act as communication master, a.s.o).

8.7 Expected interfaces

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

8.7.1 Mandatory interfaces

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

[SWS_EthIf_00102] Definition of mandatory interfaces required by module EthIf ⌈

API Function	*Header File*	*Description*
There are no mandatory interfaces.		

⌋

8.7.2 Optional interfaces

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

[SWS_EthIf_00103] Definition of optional interfaces requested by module EthIf ⌈

API Function	*Header File*	*Description*
BswM_EthIf_PortGroupLinkStateChg	BswM_EthIf.h	Function called by EthIf to indicate the link state change of a certain Ethernet switch port group.
CanXL_GetControllerMode	CanXL.h	Obtains the communication state of the indexed controller
CanXL_GetPhysAddr	CanXL.h	Obtains the physical source address used by the indexed controller
CanXL_ImmediateTransmit (draft)	CanXL.h	Request transmission of an Ethernet frame, where each upper layer a header part as element of a single linked list. All headers together with the payload form an entire Ethernet frame **Tags:** atp.Status=draft
CanXL_ProvideTxBuffer	CanXL.h	Provides access to a transmit buffer of the queue related to the specified priority
CanXL_Receive	CanXL.h	Receive a frame from the related queue.
CanXL_ReleaseRxBuffer (draft)	CanXL.h	Indication from the upper layer to release the reception buffer (ingress queue element) of the given physical Ethernet controller. **Tags:** atp.Status=draft
CanXL_SetControllerMode	CanXL.h	Enables / Disables Rx/Tx communication of the indexed controller
CanXL_Transmit	CanXL.h	Triggers transmission of a previously filled transmit buffer
CanXL_TxConfirmation	CanXL.h	Triggers frame transmission confirmation
CanXLTrcv_GetLinkState	CanXLTrcv.h	Obtains the link state of the indexed transceiver
CanXLTrcv_GetTransceiverMode	CanXLTrcv.h	Obtains the state of the indexed transceiver
CanXLTrcv_SetTransceiverMode	CanXLTrcv.h	Enables / disables the indexed transceiver
CV2x_GetBufCV2xPC5RxParams (draft)	CV2x.h	Read out values related to a received packet. For example, this could be CBR to one single packet.This API is valid only within the context of CV2x_Receive **Tags:** atp.Status=draft

▽

△

API Function	Header File	Description
CV2x_GetBufCV2xPC5TxParams (draft)	CV2x.h	Read out values related to the receive direction for a transmitted packet. For example, this could be transaction ID to one single packet. This API is valid only within the context of CV2x_TxConfirmation **Tags:** atp.Status=draft
CV2x_GetChanCV2xPC5TxParams (draft)	CV2x.h	Read values related to the receive direction of the channel. For example, this could be a Channel Busy Ratio (CBR) **Tags:** atp.Status=draft
CV2x_SetBufCV2xPC5TxParams (draft)	CV2x.h	Set values related to the transmit direction for a specific buffer (packet to be sent). For example, this can be PPPP belonging to one single packet. **Tags:** atp.Status=draft
Eth_GetControllerMode	Eth.h	Obtains the communication state of the indexed controller
Eth_GetCurrentTimeTuple (draft)	Eth.h	Reads the time tuple of the current time of the timestamp clock and the current time of the PHC in an atomic operation. If no PHC is supported, the PHC value will be a copy of the timestamp clock value. **Tags:** atp.Status=draft
Eth_GetPhcTime (draft)	Eth.h	Returns the current time value out of the HW registers of the PHC. **Tags:** atp.Status=draft
Eth_GetPhysAddr	Eth.h	Obtains the physical source address used by the indexed controller
Eth_ImmediateTransmit (draft)	EthIf.h	Request transmission of an Ethernet frame, where each upper layer a header part as element of a single linked list. All headers together with the payload form an entire Ethernet frame **Tags:** atp.Status=draft
Eth_MacSecAddRxSa (draft)	Eth.h	Requests the Ethernet Controller Driver to create a Reception Secure Association in the Controller. The Short Secure Channel Identifier is included to support XPN configurations. **Tags:** atp.Status=draft
Eth_MacSecAddTxSa (draft)	Eth.h	Requests the Ethernet Controller Driver to create a Transmission Secure Association in the Controller. The Short Secure Channel Identifier is included to support XPN configurations. **Tags:** atp.Status=draft
Eth_MacSecDeleteRxSa (draft)	Eth.h	Request the Ethernet Transceiver Driver to remove the Reception Secure Association identified by the provided Association Number. **Tags:** atp.Status=draft
Eth_MacSecDeleteTxSa (draft)	Eth.h	Request the Ethernet Transceiver Driver to remove the Transmission Secure Association identified by the provided Association Number. **Tags:** atp.Status=draft
Eth_MacSecGetMacSecStatistics (draft)	Eth.h	Request the Ethernet Driver to provide MACsec statistics. **Tags:** atp.Status=draft
Eth_MacSecGetTxSaNextPn (draft)	Eth.h	Request the Ethernet Transceiver Driver to return the Packet Number that is used for the next packet in the given Transmission Secure Association. **Tags:** atp.Status=draft

▽

△

API Function	Header File	Description
Eth_MacSecInitRxSc	Eth.h	Requests the Ethernet Controller Driver to configure a Reception Secure Channel for the given Reception Secure Channel Identifier.
Eth_MacSecResetRxSc (draft)	Eth.h	Requests the Ethernet Controller Driver to reset to MACsec default values of the Reception Secure Channel for the given Reception Secure Channel Identifier. **Tags:** atp.Status=draft
Eth_MacSecUpdateRxSa (draft)	Eth.h	Requests the Ethernet Controller Driver to update the Reception Secure Association with the given Packet Number. The Active parameter is included to change the specified AN status. **Tags:** atp.Status=draft
Eth_MacSecUpdateSecY (draft)	Eth.h	Requests the Ethernet controller to update the Sec Y/PAC at this Ethernet controller with the provided parameters. A Transmission Secure Channel with the provided SCI shall be configured during the first call. A pointer to a MACsec Basic Parameters Configuration file shall be provided to create the Secure Channel. **Tags:** atp.Status=draft
Eth_MacSecUpdateTxSa (draft)	Eth.h	Requests the Ethernet Controller Driver to update the Transmission Secure Association with the given Packet Number. The Active parameter is included to change the specified AN status. **Tags:** atp.Status=draft
Eth_ProvideTxBuffer	Eth.h	Provides access to a transmit buffer of the queue related to the specified priority
Eth_ReadMii	Eth.h	Reads a transceiver register.
Eth_Receive	Eth.h	Receive a frame from the related queue.
Eth_ReleaseRxBuffer (draft)	EthIf.h	Indication from the upper layer to release the reception buffer (ingress queue element) of the given physical Ethernet controller. **Tags:** atp.Status=draft
Eth_SetControllerMode	Eth.h	Enables / Disables Rx/Tx communication of the indexed controller
Eth_SetPhcCorrection (draft)	Eth.h	Sets PHC parameters to adapt rate and offset of the PHC. **Tags:** atp.Status=draft
Eth_SetPhcTime (draft)	Eth.h	Sets the absolute time of the PHC. **Tags:** atp.Status=draft
Eth_SetPpsSignalMode (draft)	Eth.h	Enables/disables the generation of a PPS signal **Tags:** atp.Status=draft
Eth_Transmit	Eth.h	Triggers transmission of a previously filled transmit buffer
Eth_TxConfirmation	Eth.h	Triggers frame transmission confirmation
Eth_UpdatePhysAddrFilter	Eth.h	Update the physical source address to/from the indexed controller filter. If the controller is not capable to do the filtering, the software has to do this.
Eth_WriteMii	Eth.h	Writes a transceiver register.
EthSM_CtrlModeIndication	EthSM.h	Called when mode has been read out. Either triggered by previous EthIf_GetControllerMode or by EthIf_SetControllerMode call. Can directly be called within the trigger functions.

▽

△

API Function	Header File	Description
EthSM_SleepIndication (draft)	EthSM.h	This API is called by the EthIf and indicate that a sleep indication was detected on the network. This API is only called if the ECU is acting as a passive communication slave on the corresponding communication channel (the referenced EthTrcv of the affected EthIfTransceiver has set EthTrcvActAs SlavePassiveEnabled to TRUE). This could be used e.g. for Ethernet hardware which is compliant to the OA TC10. In this case the Ethernet hardware detect an Sleep.Indication which was triggered by a Sleep.Request of the connected link partner. **Tags:** atp.Status=draft
EthSM_TrcvLinkStateChg	EthSM.h	This service is called by the Ethernet Interface to report a transceiver link state change.
EthSwt_EthRxFinishedIndication	EthSwt_Eth.h	Function inspects the Ethernet frame passed by the data pointer for management information and stores it for later use in EthSwt_EthRxFinishedIndication().
EthSwt_EthRxProcessFrame	EthSwt_Eth.h	Function inspects the Ethernet frame passed by the data pointer for management information and stores it for later use in the corresponding EthRxFinished Indication call.
EthSwt_EthTxAdaptBufferLength	EthSwt_Eth.h	Function inspects the Ethernet frame passed by the data pointer for management information and stores it for later use in EthSwt_EthRxFinishedIndication().
EthSwt_EthTxFinishedIndication	EthSwt_Eth.h	Function inspects the Ethernet frame passed by the data pointer for management information and stores it for later use in EthSwt_EthRxFinishedIndication().
EthSwt_EthTxPrepareFrame	EthSwt_Eth.h	Function inspects the Ethernet frame passed by the data pointer for management information and stores it for later use in EthSwt_EthRxFinishedIndication().
EthSwt_EthTxProcessFrame	EthSwt_Eth.h	Function inspects the Ethernet frame passed by the data pointer for management information and stores it for later use in EthSwt_EthRxFinishedIndication().
EthSwt_ExtractStreamHandleIdx (draft)	EthSwt.h	Extracts the StreamHandleIdx from the switch vendor specific part of the network packet header **Tags:** atp.Status=draft
EthSwt_GetStreamStatistics (draft)	EthSwt.h	Requests the statistics (bucket counter values) of an Ethernet switch of all configured streams. **Tags:** atp.Status=draft
EthSwt_MacSecAddRxSa (draft)	EthSwt.h	Request the Ethernet Switch Driver to create a Reception Secure Association in the Transceiver. The Short Secure Channel Identifier is included to support XPN configurations. **Tags:** atp.Status=draft
EthSwt_MacSecAddTxSa (draft)	EthSwt.h	Requests the Ethernet Switch Driver to create a Transmission Secure Association in the Transceiver. The Short Secure Channel Identifier is included to support XPN configurations. **Tags:** atp.Status=draft
EthSwt_MacSecDeleteRxSa (draft)	EthSwt.h	Request the Ethernet Switch Driver to remove the Reception Secure Association identified by the provided Association Number. **Tags:** atp.Status=draft
EthSwt_MacSecDeleteTxSa (draft)	EthSwt.h	Request the Ethernet Switch Driver to remove the Transmission Secure Association identified by the provided Association Number. **Tags:** atp.Status=draft

▽

△

API Function	*Header File*	*Description*
EthSwt_MacSecGetMacSecStatistics (draft)	EthSwt.h	Request the Ethernet Switch Driver to provide MACsec statistics. **Tags:** atp.Status=draft
EthSwt_MacSecGetTxSaNextPn (draft)	EthSwt.h	Request the Ethernet Switch Driver to return the Packet Number that is used for the next packet in the given Transmission Secure Association. **Tags:** atp.Status=draft
EthSwt_MacSecInitRxSc (draft)	EthSwt.h	Requests the Ethernet Switch Driver to configure a Reception Secure Channel for the given Secure Channel Identifier. **Tags:** atp.Status=draft
EthSwt_MacSecResetRxSc (draft)	EthSwt.h	Requests the Ethernet Switch Driver to reset to default the MACsec values of the Reception Secure Channel for the given Secure Channel Identifier. **Tags:** atp.Status=draft
EthSwt_MacSecUpdateRxSa (draft)	EthSwt.h	Request the Ethernet Switch Driver to update the Reception Secure Association with the given Packet Number. The Active parameter is included to change the specified AN status. **Tags:** atp.Status=draft
EthSwt_MacSecUpdateSecY (draft)	EthSwt.h	Requests the Ethernet Switch to update the SecY/ PAC of the PHY with the provided parameters. A Transmission Secure Channel with the provided SCI shall be configured during the first call. A pointer to a MACsec Basic Parameters Configuration file shall be provided to create the Secure Channel. **Tags:** atp.Status=draft
EthSwt_MacSecUpdateTxSa (draft)	EthSwt.h	Requests the Ethernet Switch Driver to update the Transmission Secure Association with the given Packet Number. The Active parameter is included to change the specified AN status. **Tags:** atp.Status=draft
EthSwt_PortEnableTimeStamp	EthSwt.h	Activates egress time stamping on a dedicated message object on a dedicated port of a Switch if EthSwtPortTimeStampSupport is set to TRUE for this port. The selective activation of dedicated message objects for time stamping reduces the number of notification calls only to the required calls. Some HW does store once the egress time stamp marker and some HW needs it always before transmission. There will be no disabled functionality, due to the fact, that the message type is always "time stamped" by network design.
EthSwt_SetMgmtInfo	EthSwt.h	Extends the Ethernet frame prepared previously by EthSwt_EthTxPrepareFrame() with the management information to achieve transmission only on specific ports.
EthSwt_SetStreamState (draft)	EthSwt.h	This function is called by an upper layer application (e.g. diagnostic application) via the EthIf module to control the activity status of a configured stream (given with StreamHandleIdx) within an dedicated Ethernet switch (given with SwitchIdx). **Tags:** atp.Status=draft
EthTrcv_GetBaudRate	EthTrcv.h	Obtains the baud rate of the indexed transceiver
EthTrcv_GetDuplexMode	EthTrcv.h	Obtains the duplex mode of the indexed transceiver
EthTrcv_GetLinkState	EthTrcv.h	Obtains the link state of the indexed transceiver
EthTrcv_GetTransceiverMode	EthTrcv.h	Obtains the state of the indexed transceiver

▽

△

API Function	Header File	Description
EthTrcv_MacSecAddRxSa (draft)	EthTrcv.h	Request the Ethernet Transceiver Driver to create a Reception Secure Association in the Transceiver. The Short Secure Channel Identifier is included to support XPN configurations. **Tags:** atp.Status=draft
EthTrcv_MacSecAddTxSa (draft)	EthTrcv.h	Requests the Ethernet Transceiver Driver to create a Transmission Secure Association in the Transceiver. The Short Secure Channel Identifier is included to support XPN configurations. **Tags:** atp.Status=draft
EthTrcv_MacSecDeleteRxSa (draft)	EthTrcv.h	Request the Ethernet Transceiver Driver to remove the Reception Secure Association identified by the provided Association Number. **Tags:** atp.Status=draft
EthTrcv_MacSecDeleteTxSa (draft)	EthTrcv.h	Request the Ethernet Transceiver Driver to remove the Transmission Secure Association identified by the provided Association Number. **Tags:** atp.Status=draft
EthTrcv_MacSecGetMacSecStatistics (draft)	EthTrcv.h	Request the Ethernet Transceiver Driver to provide MACsec statistics. **Tags:** atp.Status=draft
EthTrcv_MacSecGetTxSaNextPn (draft)	EthTrcv.h	Request the Ethernet Transceiver Driver to return the Packet Number that is used for the next packet in the given Transmission Secure Association. **Tags:** atp.Status=draft
EthTrcv_MacSecInitRxSc (draft)	EthTrcv.h	Requests the Ethernet Transceiver Driver to configure a Reception Secure Channel for the given Secure Channel Identifier. **Tags:** atp.Status=draft
EthTrcv_MacSecResetRxSc (draft)	EthTrcv.h	Requests the Ethernet Transceiver Driver to reset to default the MACsec values of the Reception Secure Channel for the given Secure Channel Identifier. **Tags:** atp.Status=draft
EthTrcv_MacSecUpdateRxSa (draft)	EthTrcv.h	Request the Ethernet Transceiver Driver to update the Reception Secure Association with the given Packet Number. The Active parameter is included to change the specified AN status. **Tags:** atp.Status=draft
EthTrcv_MacSecUpdateSecY (draft)	EthTrcv.h	Requests the Ethernet Transceiver to update the SecY/PAC of the PHY with the provided parameters. A Transmission Secure Channel with the provided SCI shall be configured during the first call. A pointer to a MACsec Basic Parameters Configuration file shall be provided to create the Secure Channel. **Tags:** atp.Status=draft
EthTrcv_MacSecUpdateTxSa (draft)	EthTrcv.h	Requests the Ethernet Transceiver Driver to update the Transmission Secure Association with the given Packet Number. The Active parameter is included to change the specified AN status. **Tags:** atp.Status=draft
EthTrcv_SetTransceiverMode	EthTrcv.h	Enables / disables the indexed transceiver
EthTrcv_StartAutoNegotiation	EthTrcv.h	Restarts the negotiation of the transmission parameters used by the indexed transceiver
Fw_StreamStateIndication (draft)	Fw_Cbk.h	The function is called by the EthIf once it has successfully set the StreamHandleIdx in the switch. **Tags:** atp.Status=draft

▽

△

API Function	Header File	Description
Fw_StreamStatisticsIndication (draft)	Fw_Cbk.h	The function is called by the lower layer once it has successfully retrieved the stream statistics (i.e. bucket counter values) from the EthSwt driver given with SwitchIdx **Tags:** atp.Status=draft
LSduR_EthIfRxIndication (draft)	LSduR_EthIf.h	Indication of a received PDU from a lower layer communication interface module.
LSduR_EthIfTriggerTransmit (draft)	LSduR_EthIf.h	Within this API, the upper layer module (called module) shall check whether the available data fits into the buffer size reported by PduInfoPtr->Sdu Length. If it fits, it shall copy its data into the buffer provided by PduInfoPtr->SduDataPtr and update the length of the actual copied data in PduInfoPtr->Sdu Length. If not, it returns E_NOT_OK without changing PduInfoPtr.
LSduR_EthIfTxConfirmation (draft)	LSduR_EthIf.h	The lower layer communication interface module confirms the transmission of a PDU, or the failure to transmit a PDU.
Mka_GetMacSecStatisticsNotification (draft)	Mka.h	Callback to signal completion of call to EthIf_MacSecGetMacSecStatistics, EthIf_EthMacSecGetMacSecStatistics or EthIf_SwitchMacSecGetMacSecStatistics. **Tags:** atp.Status=draft
Mka_LinkStateChange (draft)	Mka.h	To inform MKA that a dedicated Trcv/Switch/PAC port link state has changed **Tags:** atp.Status=draft
Mka_MacSecAddRxSaNotification (draft)	Mka.h	Callback to signal completion of call to EthIf_MacSecAddRxSa, EthIf_EthMacSecAddRxSa or EthIf_SwitchMacSecAddRxSa has finished. **Tags:** atp.Status=draft
Mka_MacSecAddTxSaNotification (draft)	Mka.h	Callback to signal completion of call to EthIf_MacSecAddTxSa, EthIf_EthMacSecAddTxSa or EthIf_SwitchMacSecAddTxSa. **Tags:** atp.Status=draft
Mka_MacSecUpdateSecYNotification (draft)	Mka.h	Callback to signal completion of call to EthIf_MacSecUpdateSecY, EthIf_EthMacSecUpdateSecY or EthIf_SwitchMacSecUpdateSecY has finished. **Tags:** atp.Status=draft
WEth_GetBufWRxParams	WEth.h	Read out values related to the receive direction for a received packet. For example, this could be RSSI or Channel belonging to one single packet.This API is valid only within the context of WEth_Receive
WEth_GetBufWTxParams	WEth.h	Read out values related to the transmit direction for a transmitted packet. This API is valid only within the context of WEth_TxConfirmation.
WEth_SetBufWTxParams	WEth.h	Set values related to the transmit direction for a specific buffer (packet to be sent). For example, this can be the desired transmit power or the channel belonging to one single packet.
WEthTrcv_GetChanRxParams	WEthTrcv.h	Read values related to the receive direction of the transceiver. For example, this could be a Channel Busy Ratio (CBR) or the average Channel Idle Time (CIT).
WEthTrcv_SetChanRxParams	WEthTrcv.h	Set values related to the receive direction of a transceiver's wireless channel.For example, this could be a channel parameter like the frequency.

▽

△

API Function	Header File	Description
WEthTrcv_SetChanTxParams	WEthTrcv.h	Set values related to the transmit direction of a transceiver's wireless channel. For example, this could be the bitrate of a channel.
WEthTrcv_SetRadioParams	WEthTrcv.h	Set values related to a transceiver's wireless radio. For example, this could be the selection of the radio settings (channel, ...).

⌋

8.7.3 Configurable interfaces

In this section, all interfaces are listed where the target function could be configured. The target function is usually a callback function. The names of this kind of interfaces are not fixed because they are configurable.

Terms and definitions:

Reentrant interface is reentrant

Don't care reentrancy of interface not relevant for this module (in general it is in this case not reentrant).

[SWS_EthIf_00108] Definition of configurable interface <User>_TrcvLinkStateChg ⌈

Service Name	<User>_TrcvLinkStateChg	
Syntax	`void <User>_TrcvLinkStateChg ( uint8 CtrlIdx, EthTrcv_LinkStateType TrcvLinkState )`	
Sync/Async	Synchronous	
Reentrancy	Don't care	
Parameters (in)	CtrlIdx	Index of the Ethernet controller within the context of the Ethernet Interface
	TrcvLinkState	ETHTRCV_LINK_STATE_DOWN transceiver link is down ETHTRCV_LINK_STATE_ACTIVE transceiver link is up
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Indicates the change of a transceiver state	
Available via	configurable	

⌋

[SWS_EthIf_00109] ⌈The callback function shall be configurable by the configuration parameter: `EthIfTrcvLinkStateChgFunction`.⌋

[SWS_EthIf_00230] ⌈Upon change of the physical link state `<User>_TrcvLinkStateChg` shall be invoked for every affected EthIfController.⌋

Note: This means that <User>_TrcvLinkStateChg can be called independent of the according EthIfController mode when more than one EthIfController is bound to the same physical channel.

9 Sequence diagrams

The sequence diagrams show the basic operations carried out during operation. They show the interaction of the Ethernet Interface with upper layer BSW module and the underlying Ethernet Controller Driver.

Please note that the sequence diagrams are an extension for illustrational purposes to ease understanding of the specification.

9.1 Initialization

Name: EthIf_Initalization
Package: EthIf
Version: 1.0
Author: fix0ec2

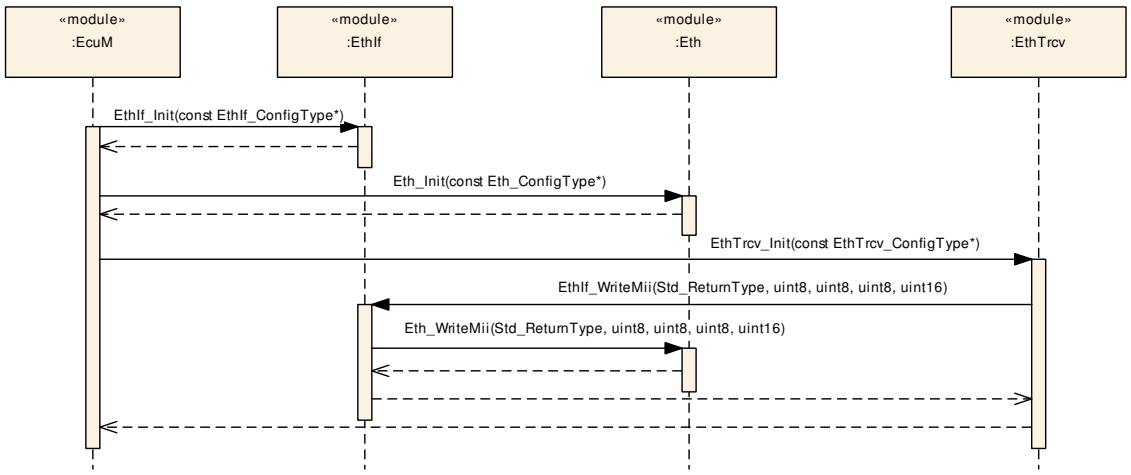

Figure 9.1: Initialization

9.2 Communication Initialization

Name: EthIf_CommunicationInitialization
Package: EthIf
Version: 1.0
Author: fix0ec2

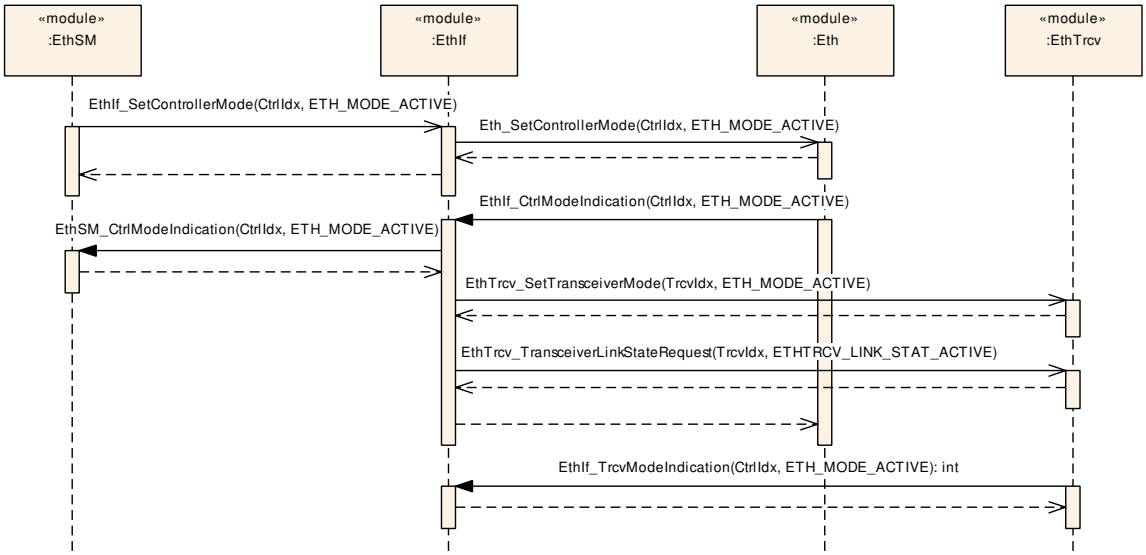

Figure 9.2: Communication Initialization

9.3 Switch Initialization

Name: EthIf_SwitchInitalization
Package: EthIf
Version: 1.0
Author: fix0ec2

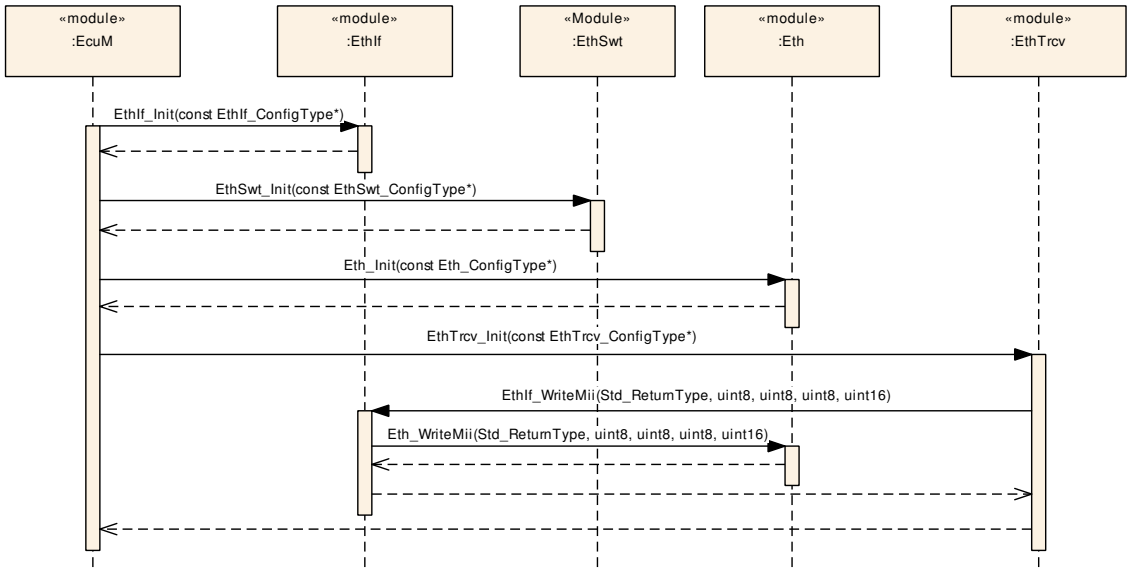

Figure 9.3: Switch Initialization

9.4 Data Transmission

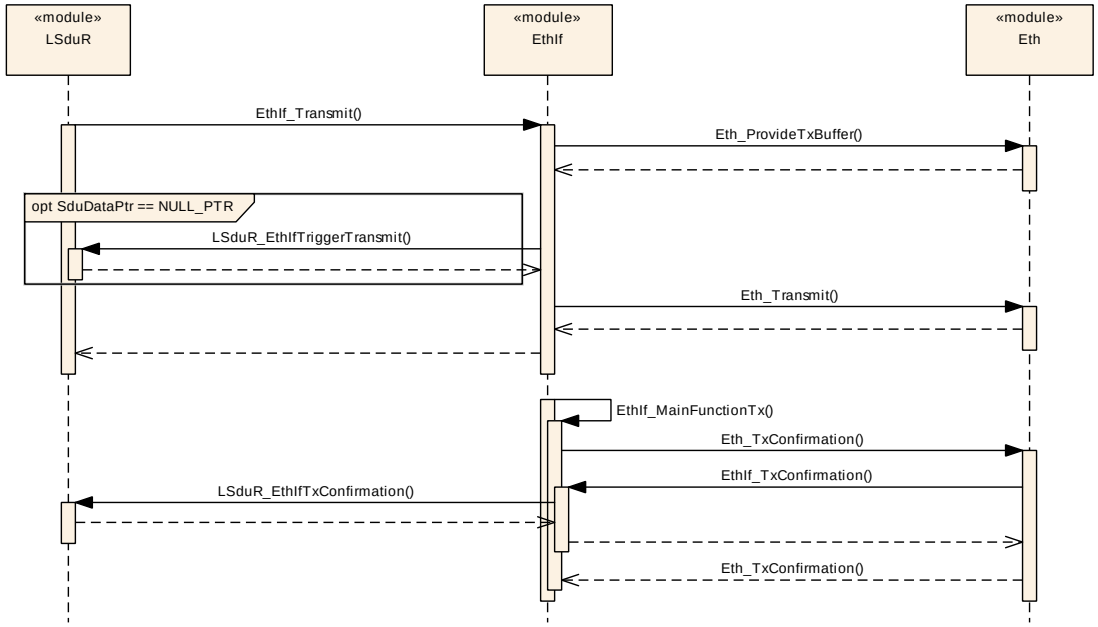

Figure 9.4: Frame Transmission in Polling Mode with indirect data provision

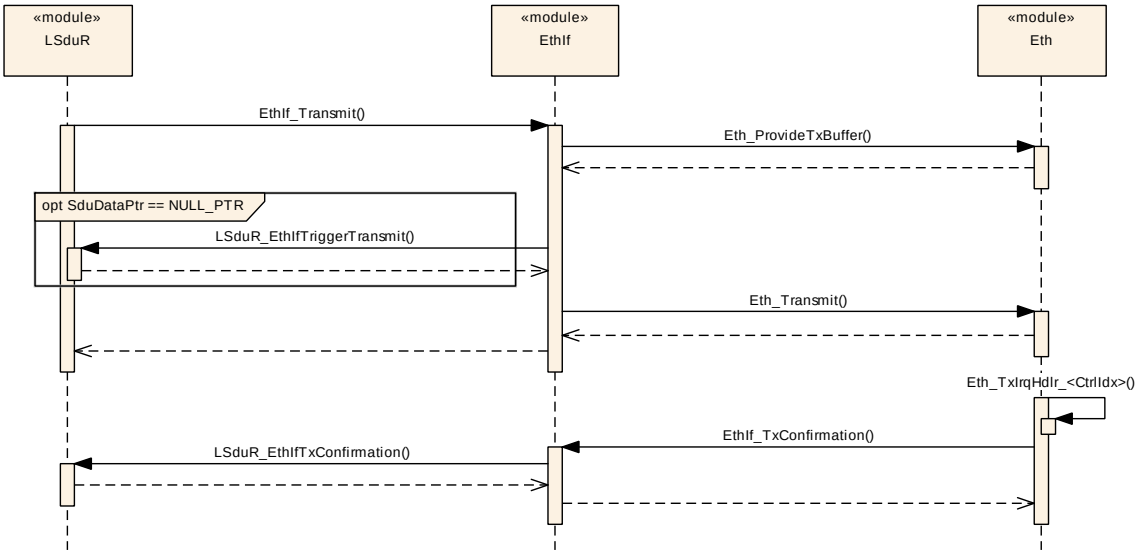

Figure 9.5: Frame Transmission in Interrupt Mode with indirect data provision

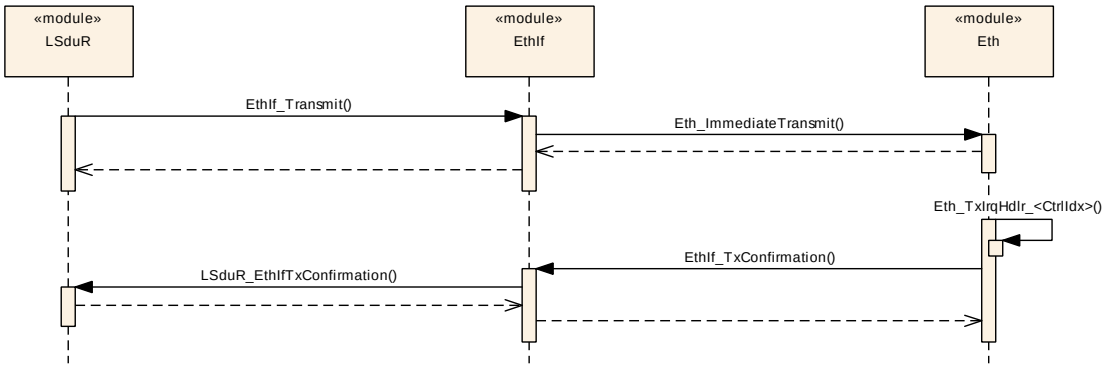

Figure 9.6: Frame Transmission in Interrupt Mode with direct data provision

9.5 Data Reception

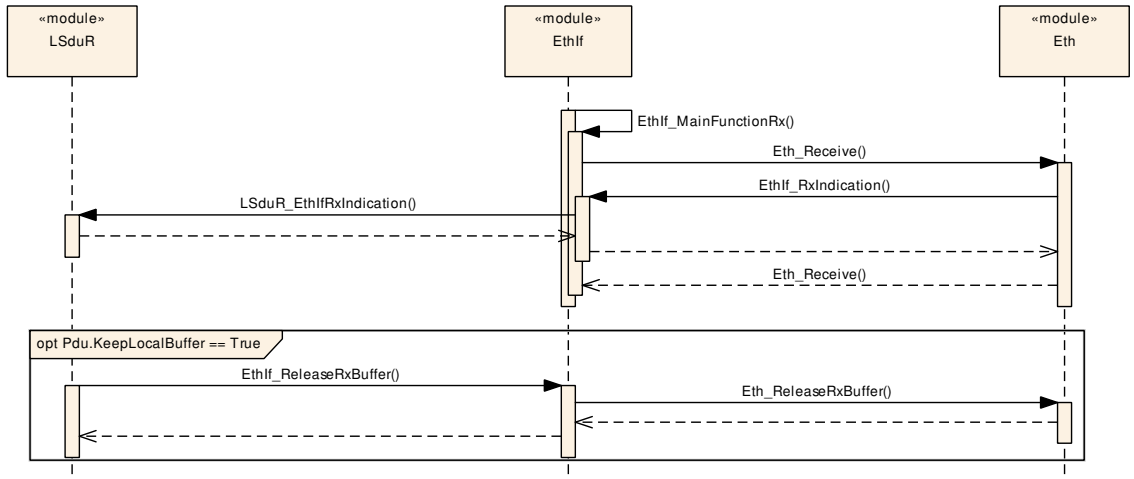

Figure 9.7: Frame Reception in Polling Mode

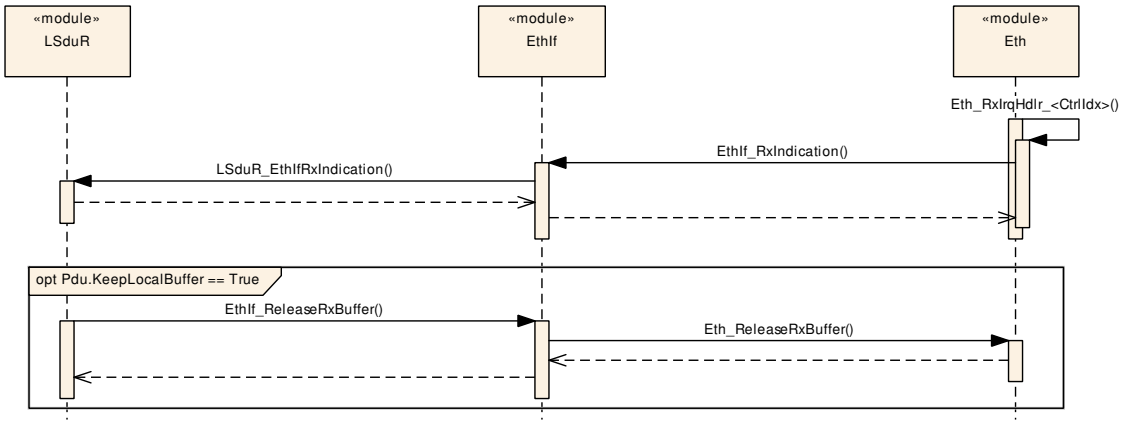

Figure 9.8: Frame Reception in Interrupt Mode

9.6 Link State Change

Name: EthIf_LinkStateChange
Package: EthIf
Version: 1.0
Author: fix0ec2

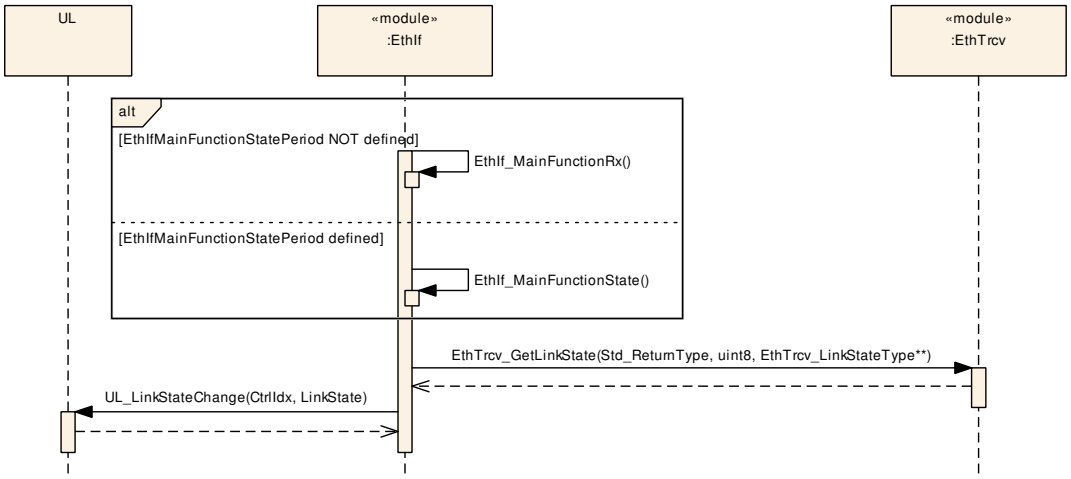

Figure 9.9: Link State Change

9.7 Link State Change without Port Groups

Name: EthIf_EthSwt_LinkStateChange_NoPortGroup
Package: EthIf
Version: 1.0
Author: fix0ec2

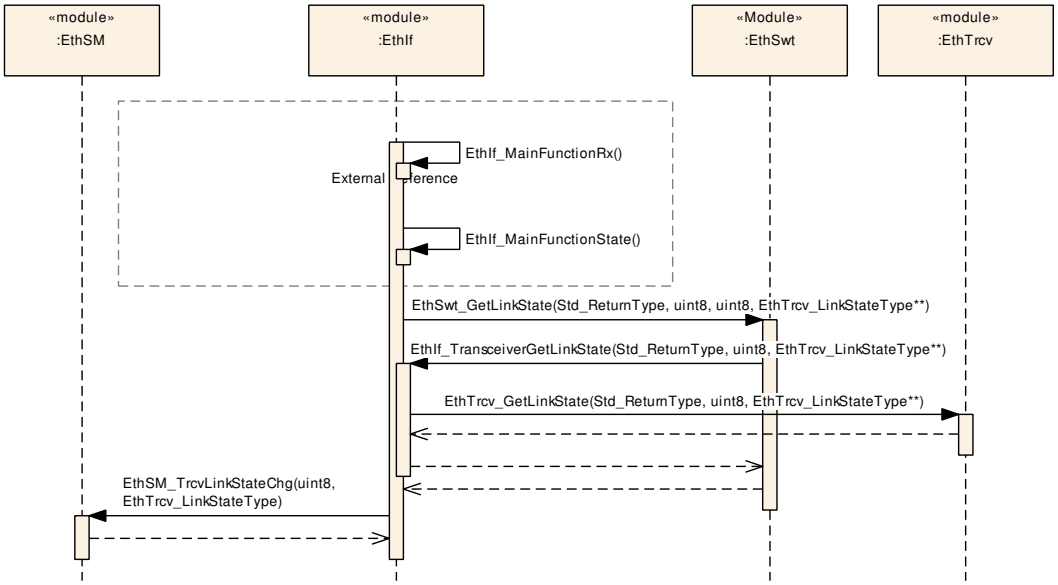

Figure 9.10: Link State Change without Port Groups

9.8 Link State Change with Port Groups

Name: EthIf_EthSwt_LinkStateChangePortGroupControl
Package: EthIf
Version: 1.0
Author: fix0ec2

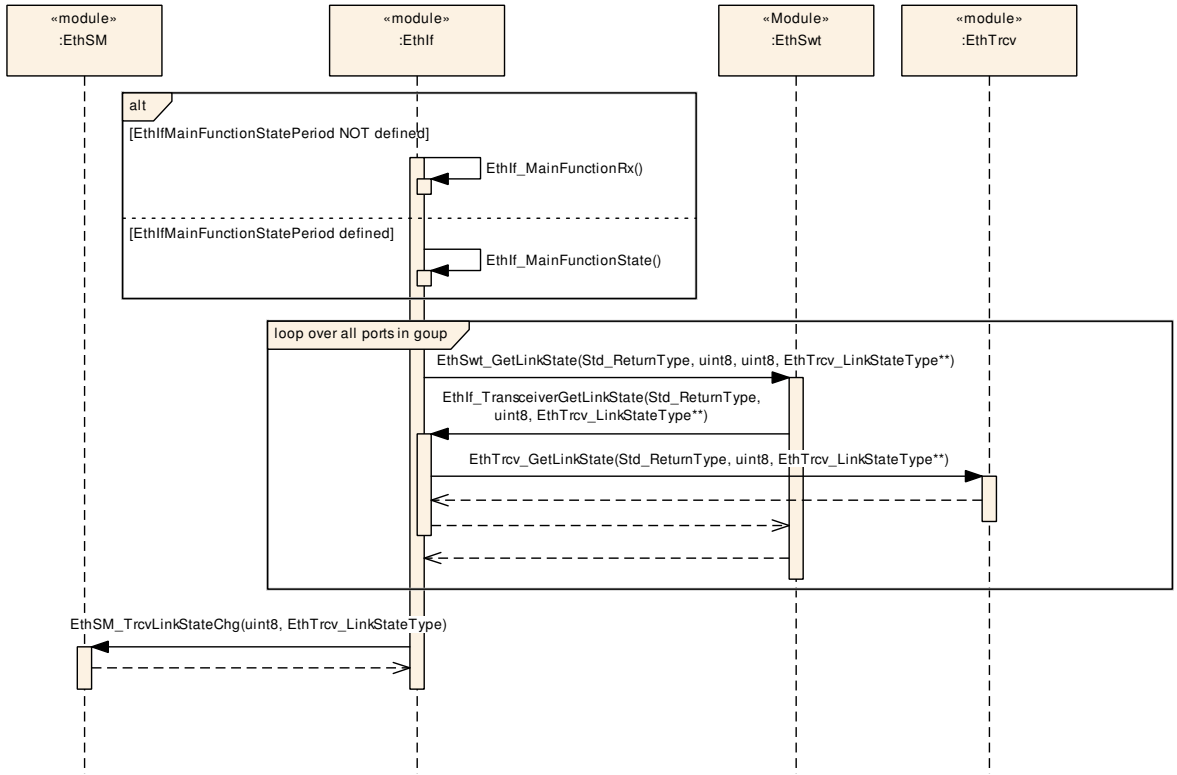

Figure 9.11: Link State Change with Port Groups

9.9 Link State Change with Port Groups and Partial Network Cluster

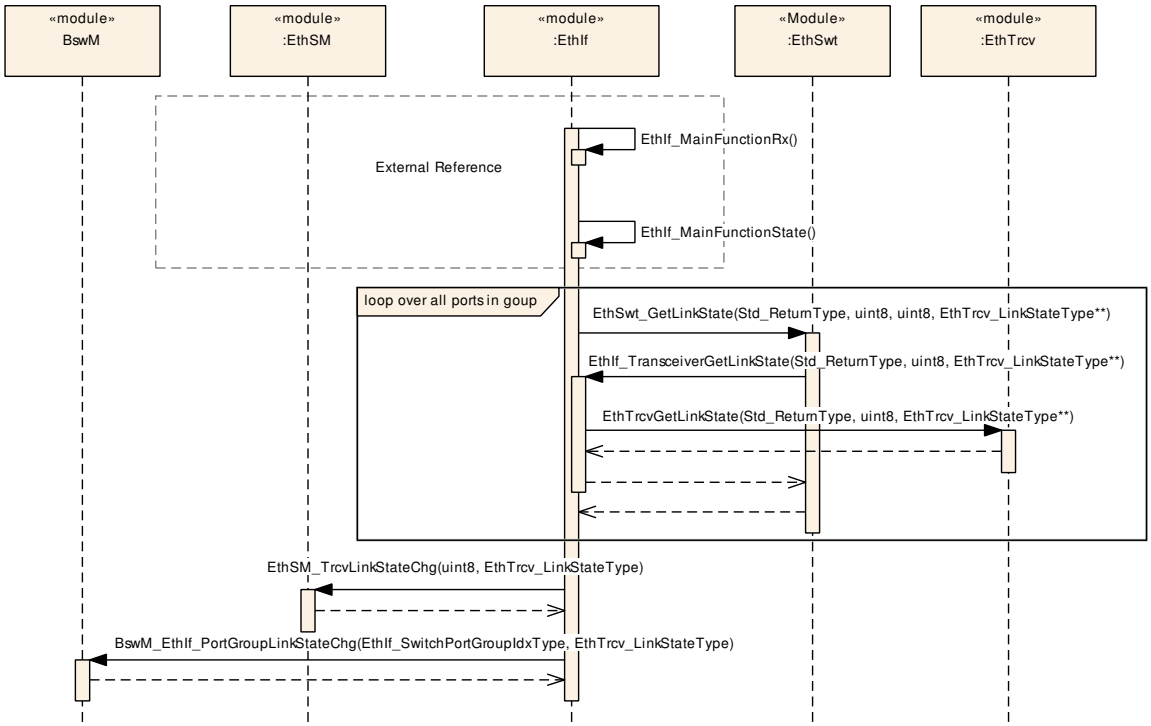

Figure 9.12: and Partial Network Cluster

9.10 Switch Management support

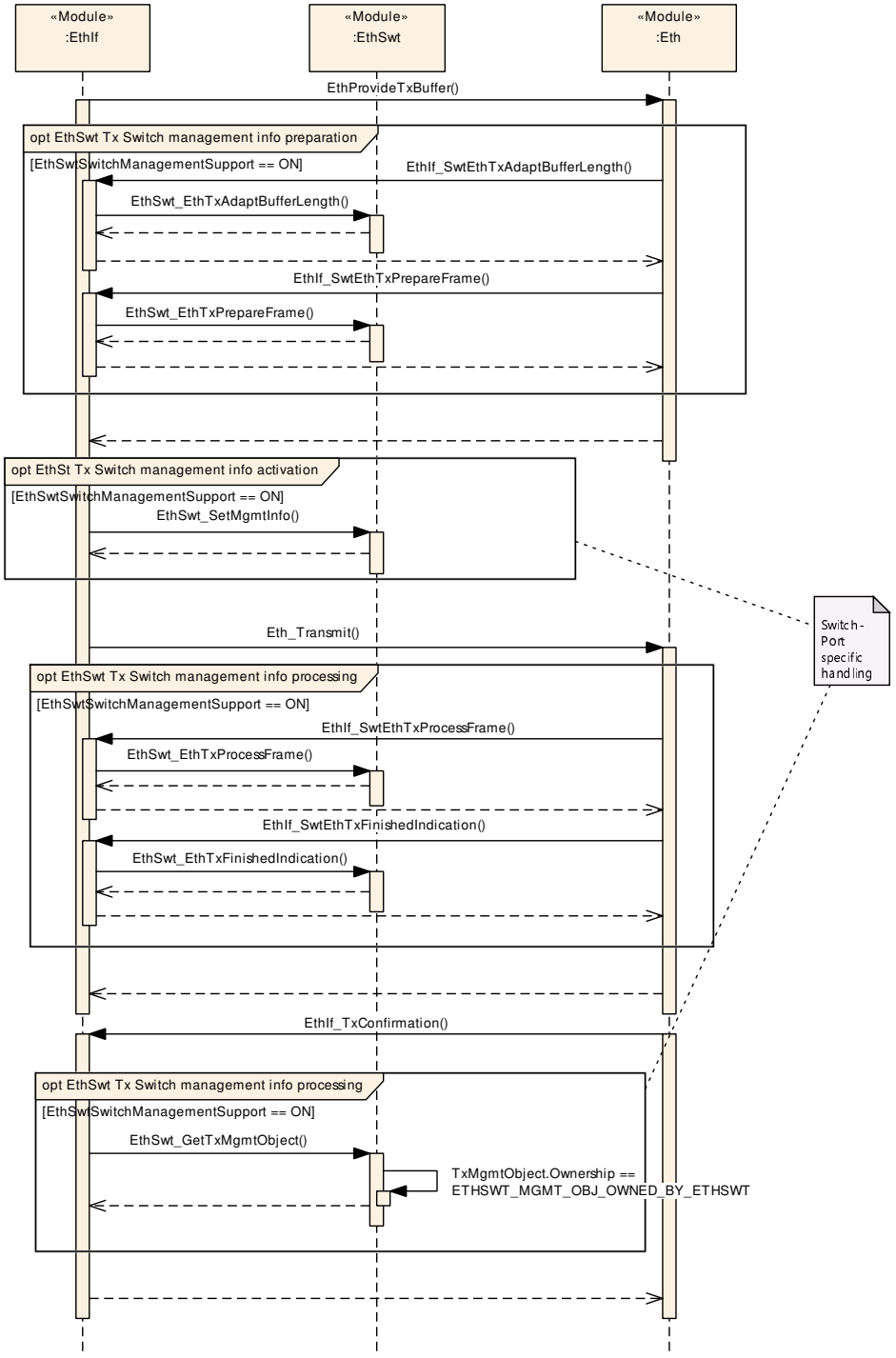

Figure 9.13: Switch Management support for transmission 1

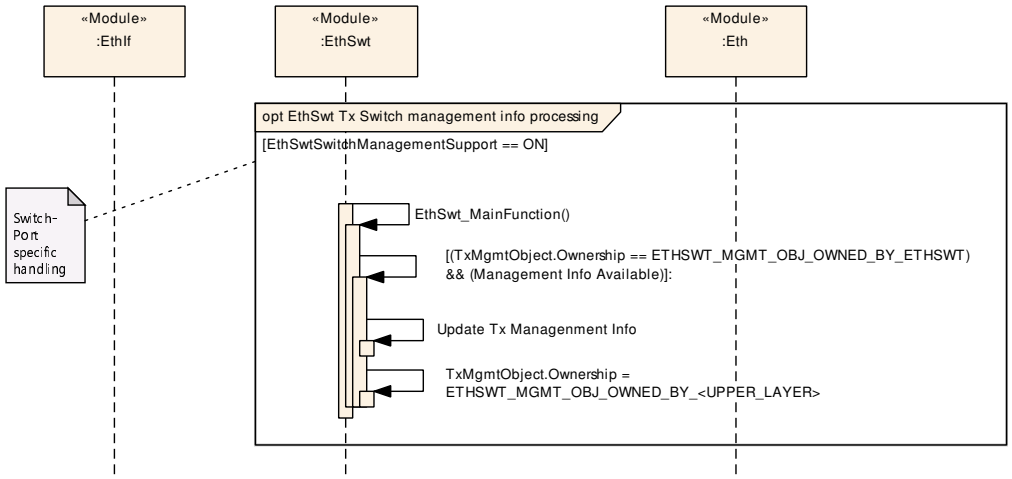

Figure 9.14: Switch Management support for transmission 2

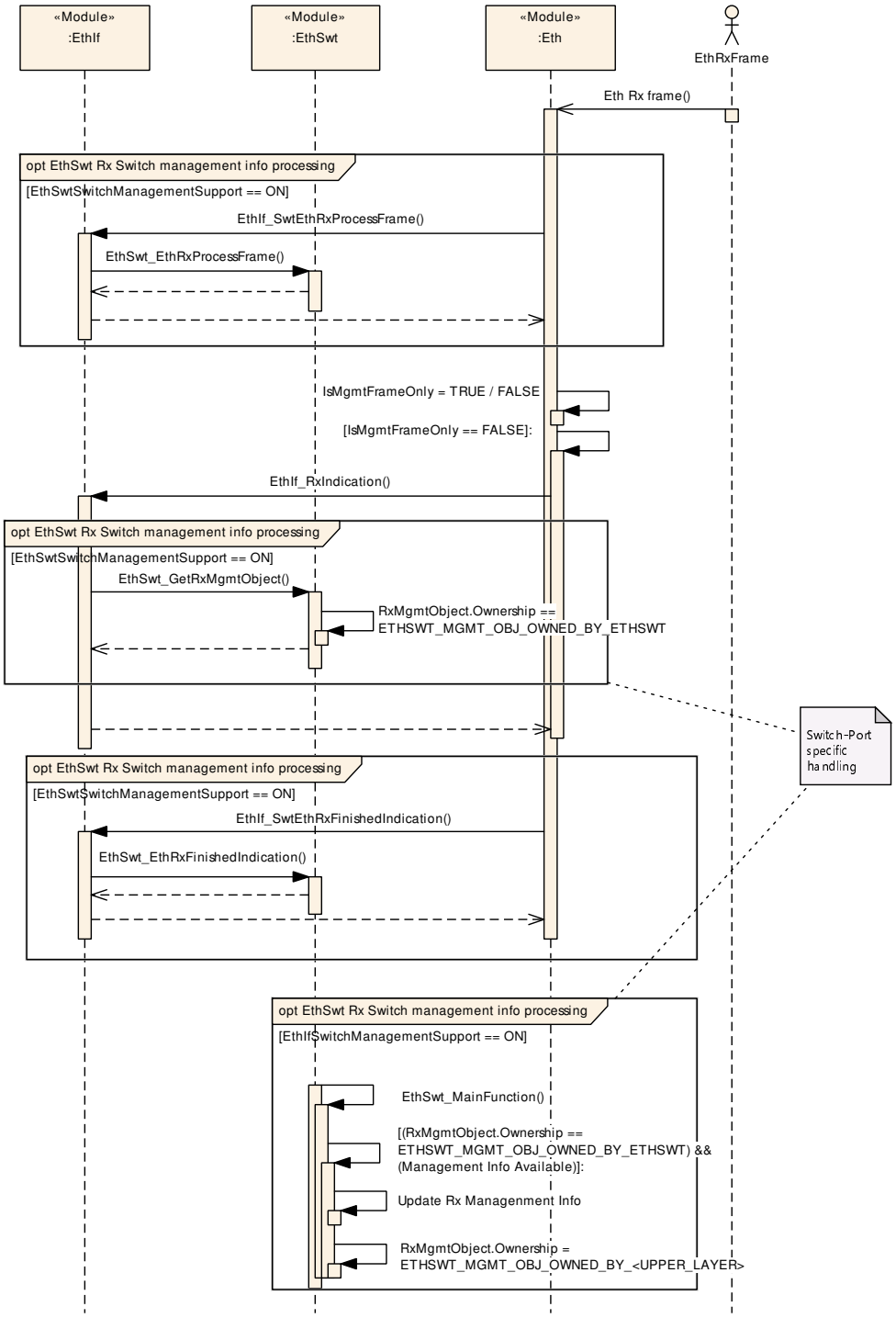

Figure 9.15: Switch Management support for reception

10 Configuration specification

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

Chapter 10.2 specifies the structure (containers) and the parameters of the module Ethernet Interface.

Chapter 10.4 specifies published information of the module Ethernet Interface.

10.1 How to read this chapter

For details refer to [6] Chapter 10.1 *"Introduction to configuration specification"*.

10.2 Containers and configuration parameters

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

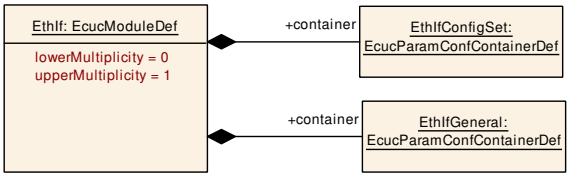

Figure 10.1: Ethernet Interface

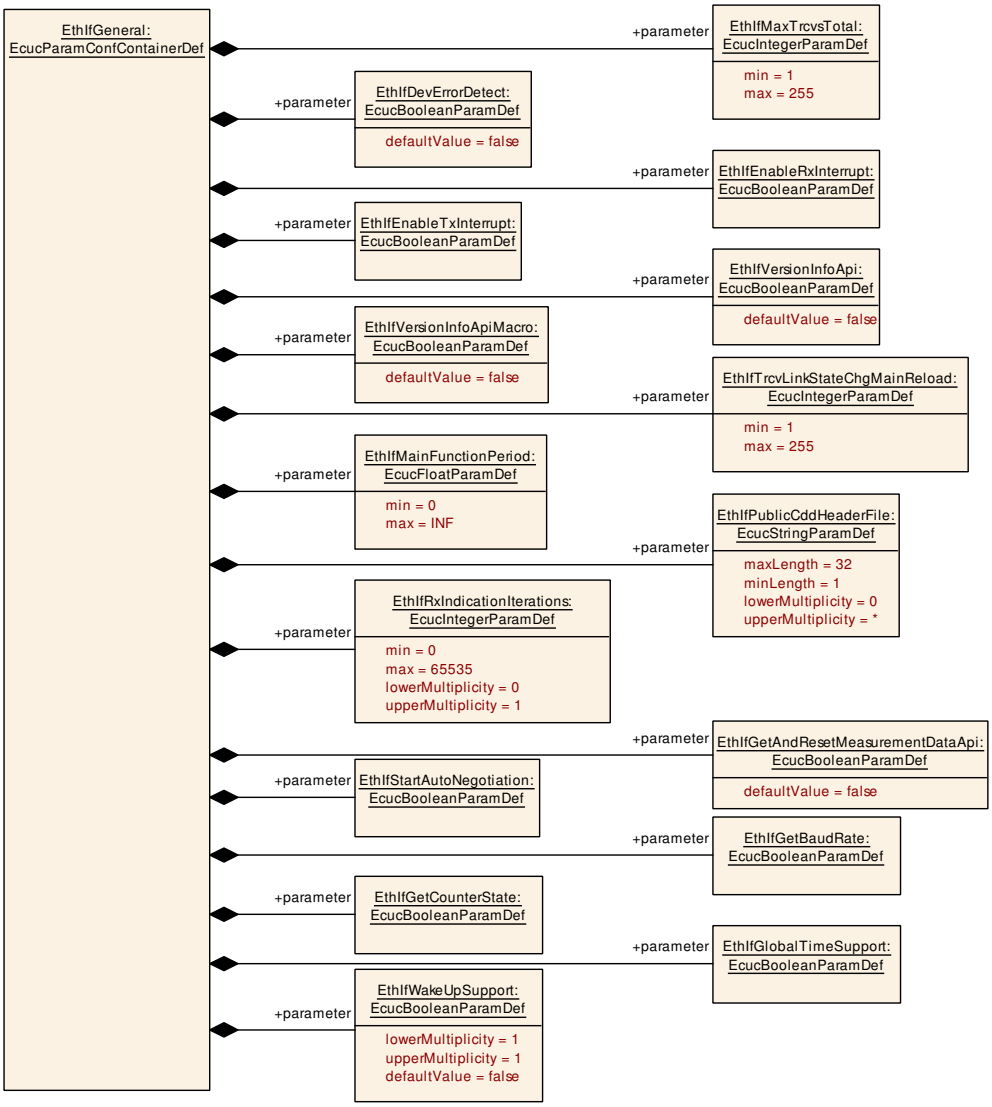

Figure 10.2: Ethernet Interface general configuration structure

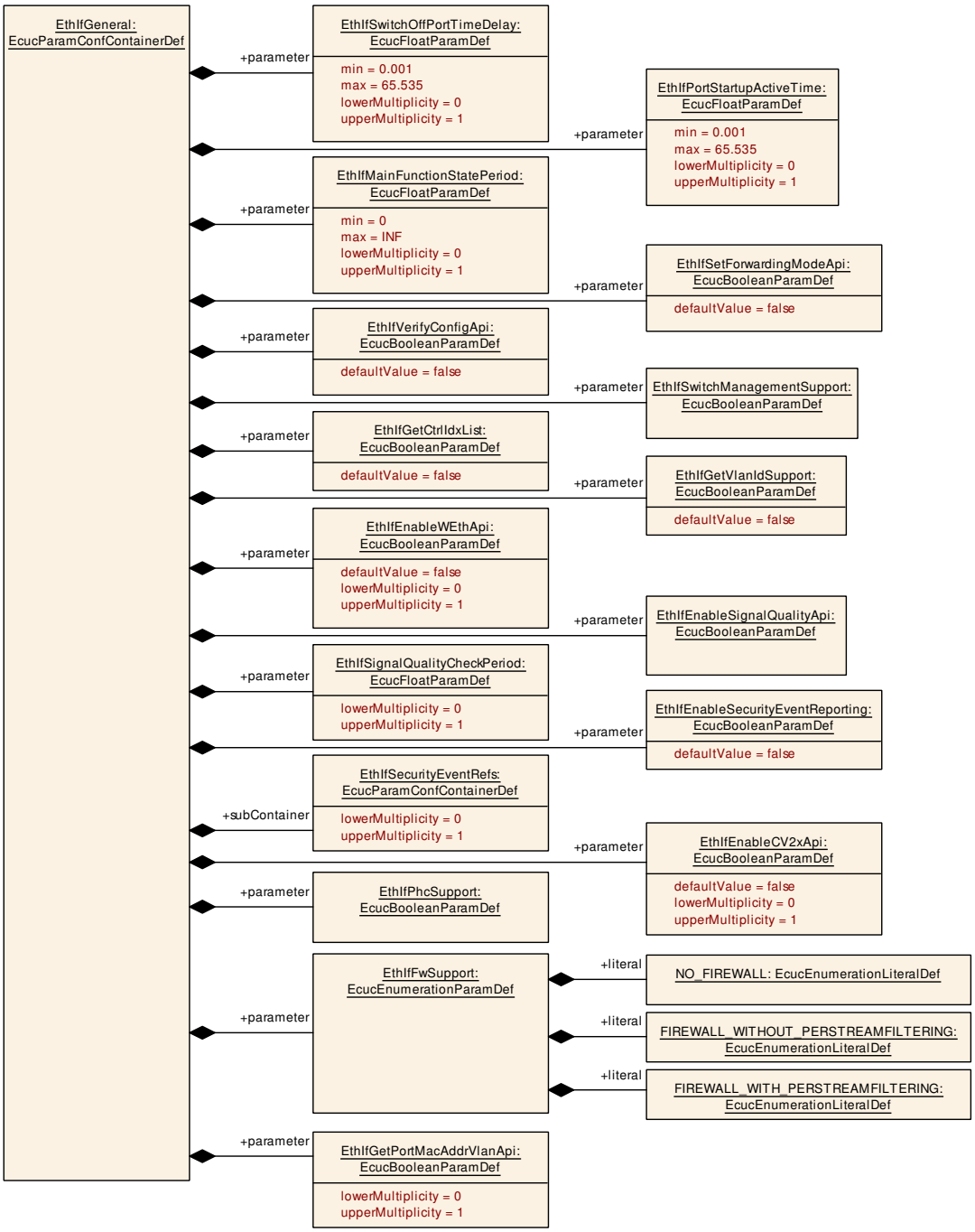

Figure 10.3: Ethernet Interface general configuration structure (continued)

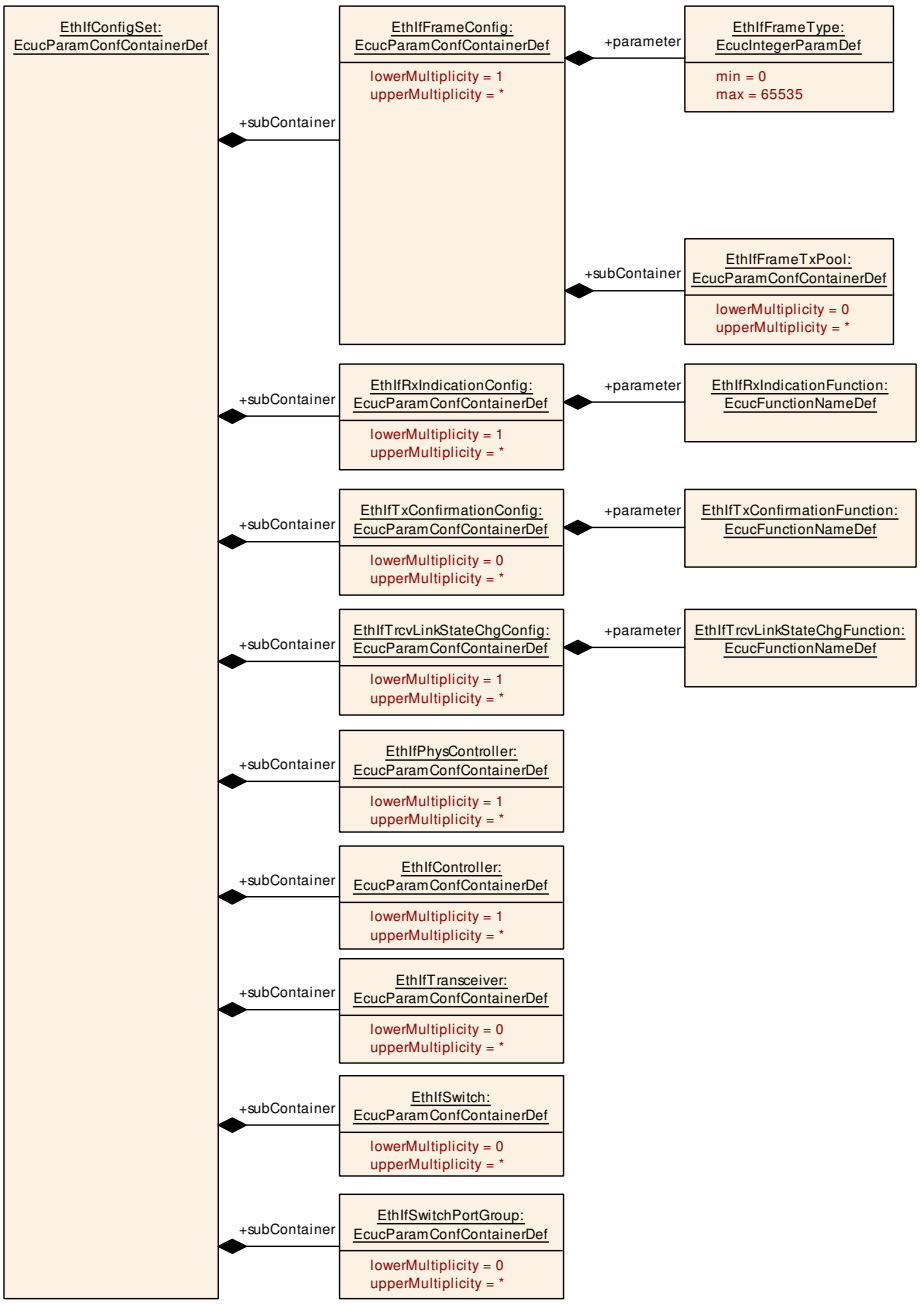

Figure 10.4: Ethernet Interface interface configuration structure

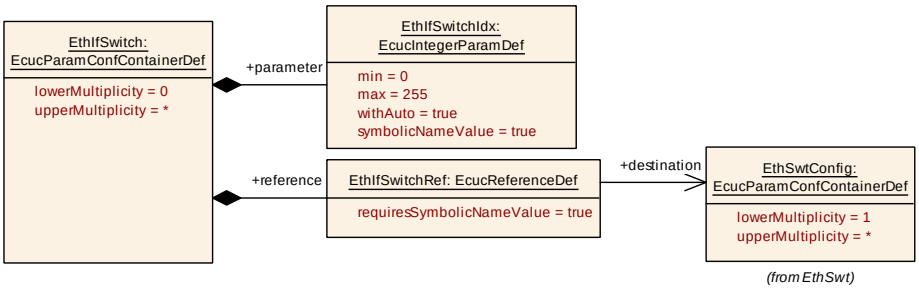

Figure 10.5: Ethernet Interface Switch configuration structure

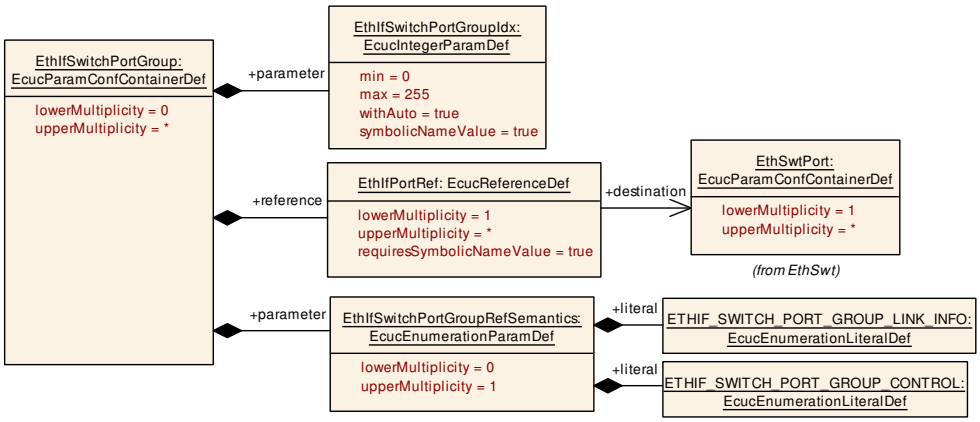

Figure 10.6: Ethernet Interface SwitchPortGroup configuration structure

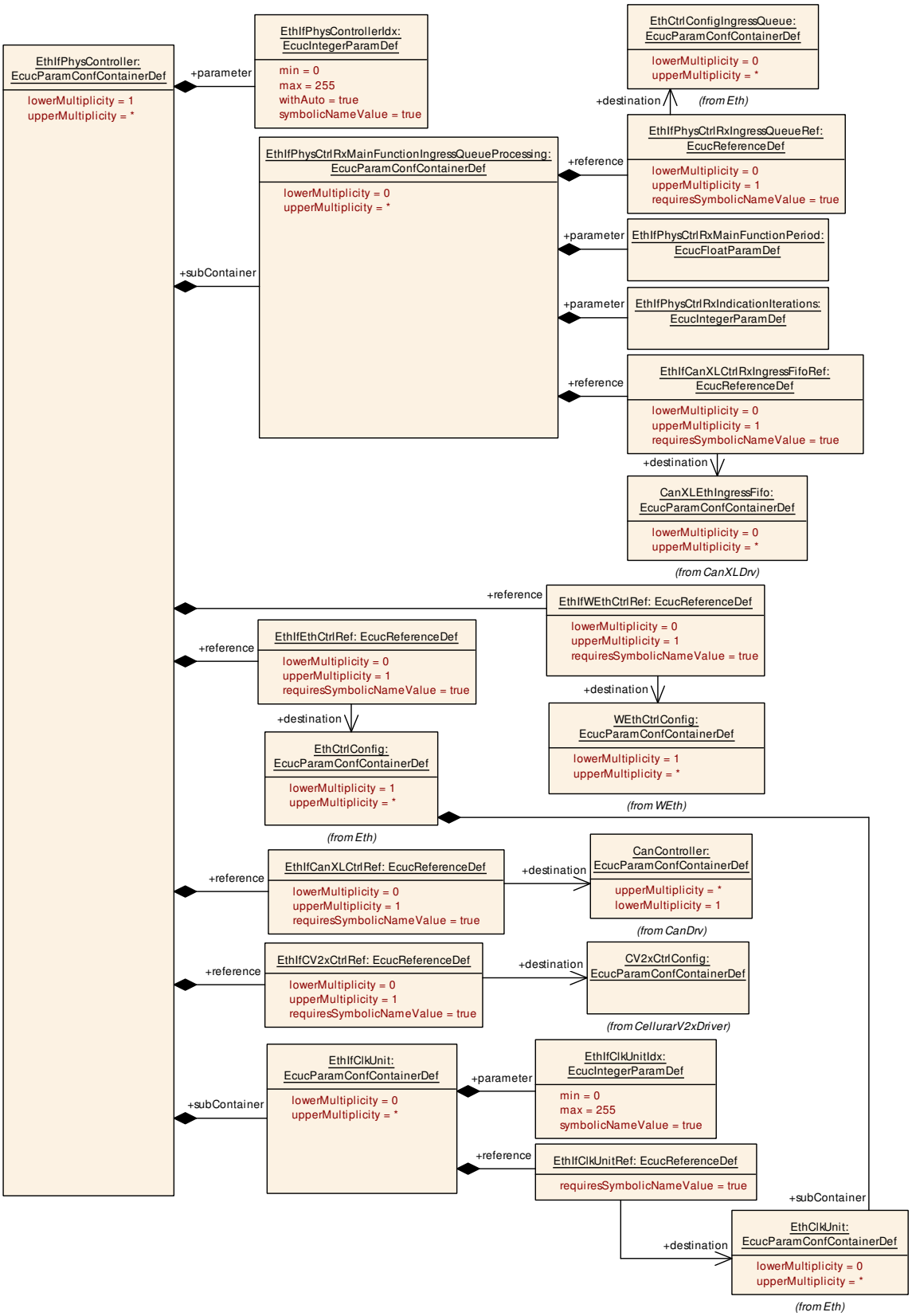

Figure 10.7: Ethernet Interface physical controller configuration structure

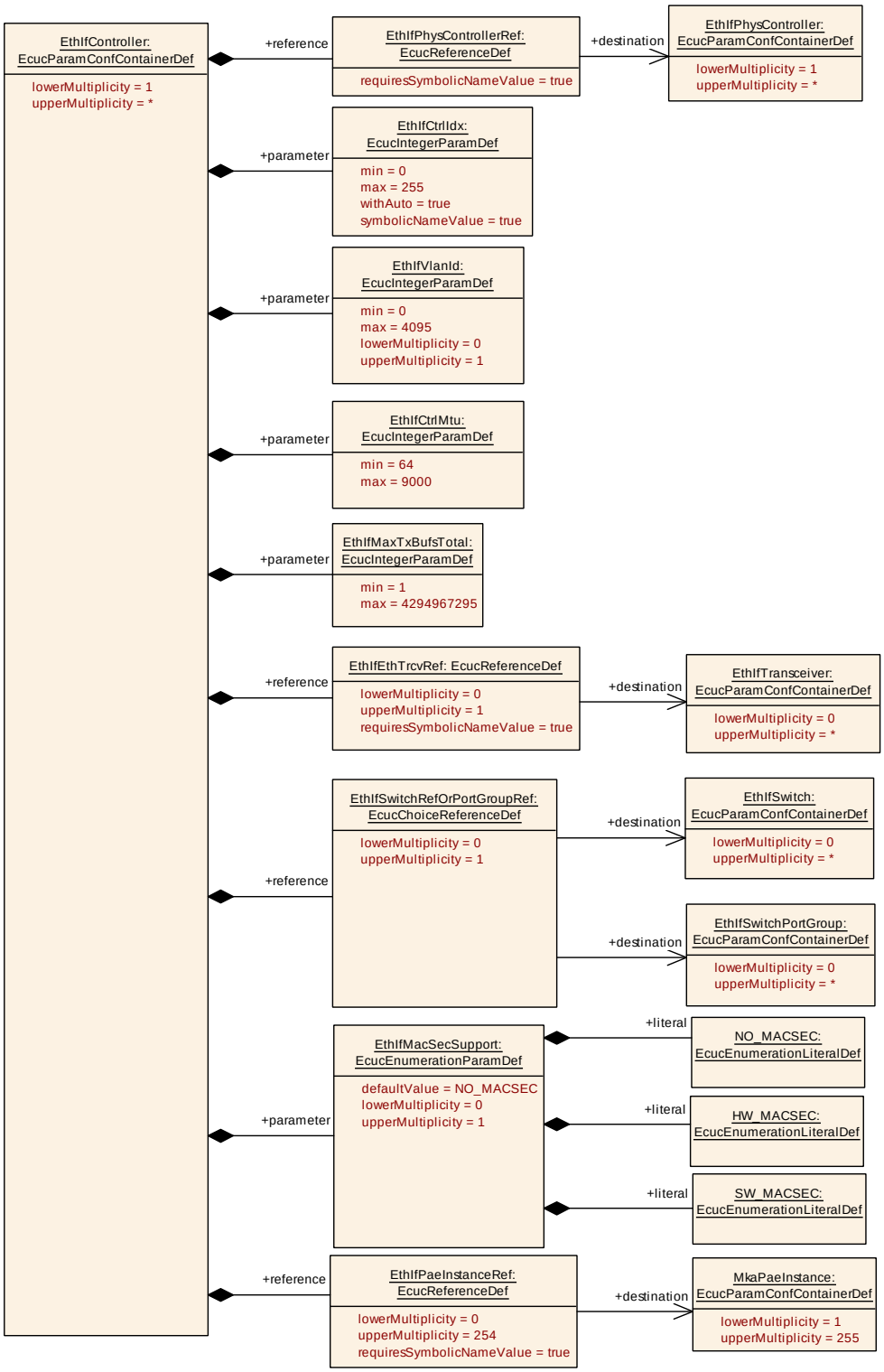

Figure 10.8: Ethernet Interface controller configuration structure

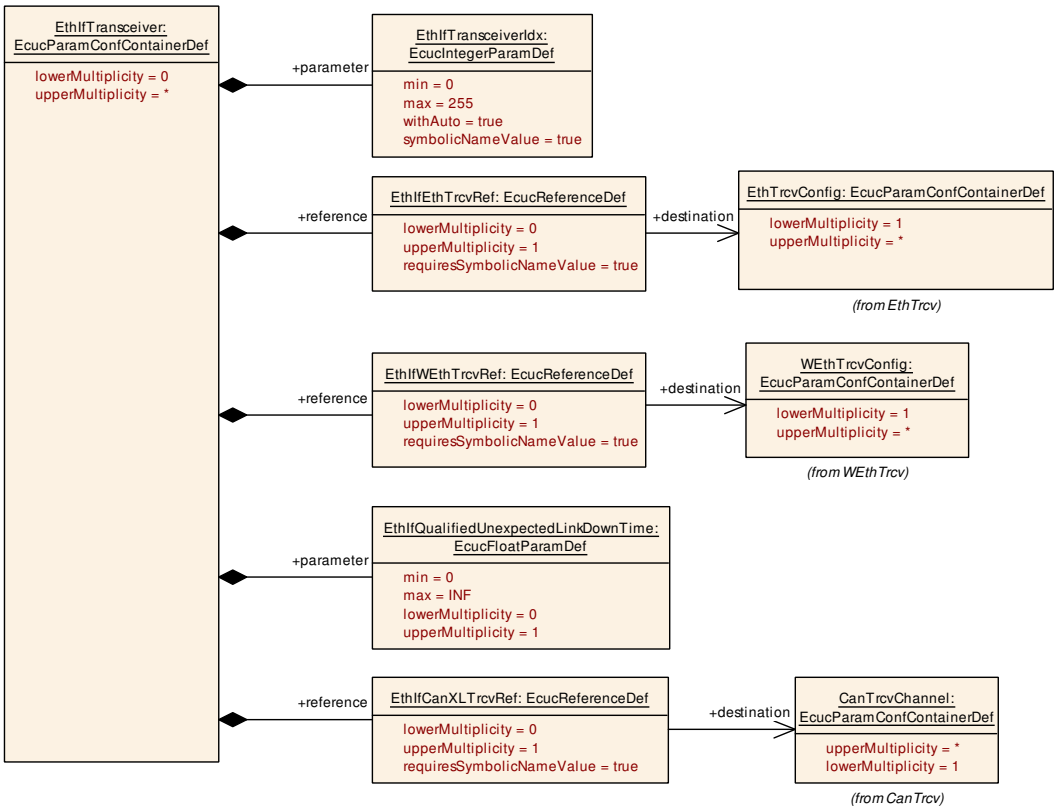

Figure 10.9: Ethernet Interface transceiver configuration structure

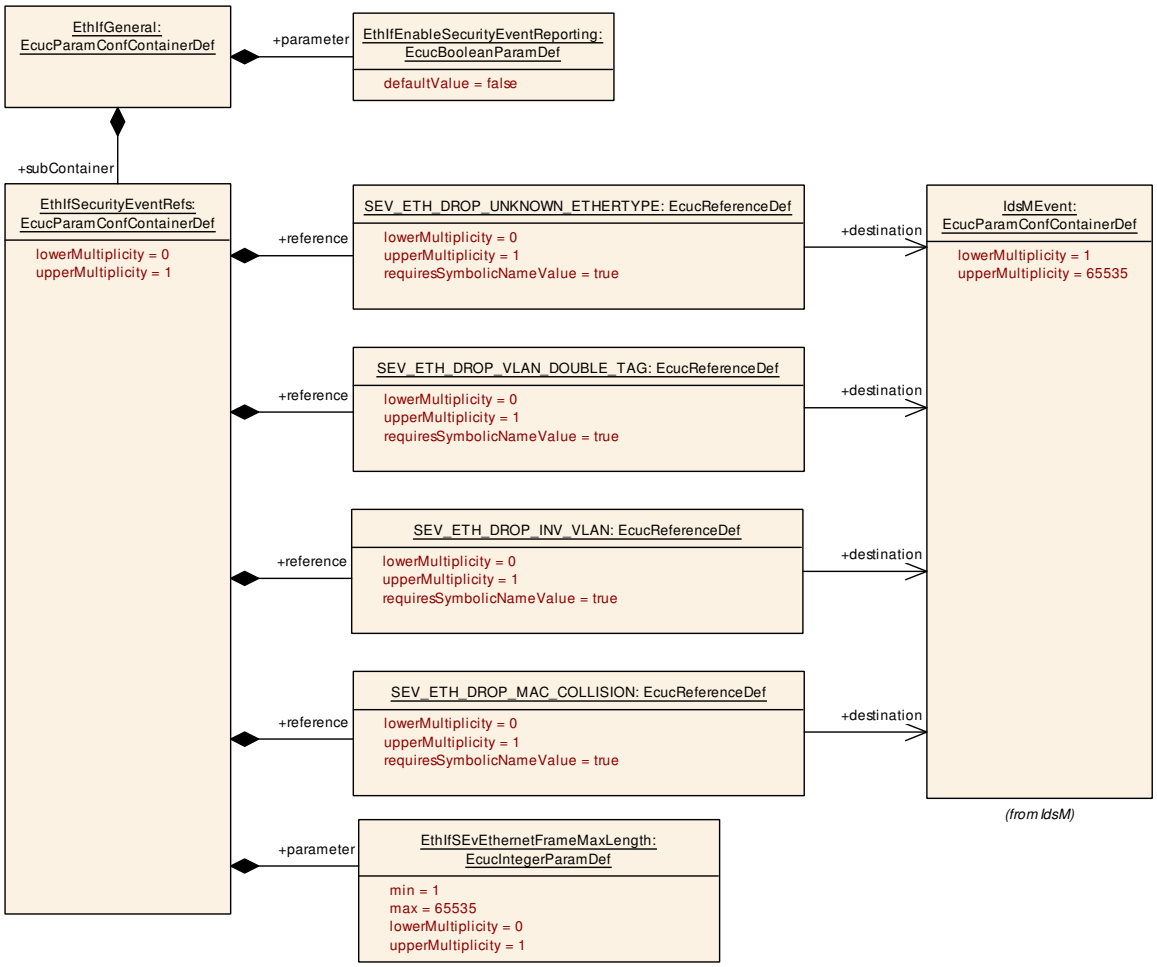

Figure 10.10: Ethernet security event ref

10.2.1 EthIf

[ECUC_EthIf_00049] Definition of EcucModuleDef EthIf ⌈

Module Name	EthIf
Description	Configuration of the EthIf (Ethernet Interface) module.
Post-Build Variant Support	true
Supported Config Variants	VARIANT-LINK-TIME, VARIANT-POST-BUILD, VARIANT-PRE-COMPILE

Included Containers		
Container Name	**Multiplicity**	**Dependency**
EthIfConfigSet	1	Collecting container for all parameters with post-build configuration classes.
EthIfGeneral	1	This container contains the general configuration parameters of the Ethernet Interface.

⌋

10.2.2 EthIfGeneral

[ECUC_EthIf_00001] Definition of EcucParamConfContainerDef EthIfGeneral ⌈

Container Name	EthIfGeneral
Parent Container	EthIf
Description	This container contains the general configuration parameters of the Ethernet Interface.
Multiplicity	1
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
EthIfDevErrorDetect	1	[ECUC_EthIf_00004]
EthIfEnableCV2xApi	0..1	[ECUC_EthIf_00091]
EthIfEnableRxInterrupt	1	[ECUC_EthIf_00005]
EthIfEnableSecurityEventReporting	1	[ECUC_EthIf_00079]
EthIfEnableSignalQualityApi	1	[ECUC_EthIf_00076]
EthIfEnableTxInterrupt	1	[ECUC_EthIf_00006]
EthIfEnableWEthApi	0..1	[ECUC_EthIf_00075]
EthIfFwSupport	1	[ECUC_EthIf_00094]
EthIfGetAndResetMeasurementDataApi	1	[ECUC_EthIf_00072]
EthIfGetBaudRate	1	[ECUC_EthIf_00034]
EthIfGetCounterState	1	[ECUC_EthIf_00035]
EthIfGetCtrlIdxList	1	[ECUC_EthIf_00070]
EthIfGetPortMacAddrVlanApi	0..1	[ECUC_EthIf_00108]
EthIfGetVlanIdSupport	1	[ECUC_EthIf_00071]
EthIfGlobalTimeSupport	1	[ECUC_EthIf_00039]
EthIfMainFunctionPeriod	1	[ECUC_EthIf_00023]
EthIfMainFunctionStatePeriod	0..1	[ECUC_EthIf_00056]
EthIfMaxTrcvsTotal	1	[ECUC_EthIf_00003]
EthIfPhcSupport	1	[ECUC_EthIf_00102]
EthIfPortStartupActiveTime	0..1	[ECUC_EthIf_00055]
EthIfPublicCddHeaderFile	0..*	[ECUC_EthIf_00024]
EthIfRxIndicationIterations	0..1	[ECUC_EthIf_00030]
EthIfSetForwardingModeApi	1	[ECUC_EthIf_00062]
EthIfSignalQualityCheckPeriod	0..1	[ECUC_EthIf_00077]
EthIfStartAutoNegotiation	1	[ECUC_EthIf_00033]
EthIfSwitchManagementSupport	1	[ECUC_EthIf_00064]
EthIfSwitchOffPortTimeDelay	0..1	[ECUC_EthIf_00054]
EthIfTrcvLinkStateChgMainReload	1	[ECUC_EthIf_00009]
EthIfVerifyConfigApi	1	[ECUC_EthIf_00063]
EthIfVersionInfoApi	1	[ECUC_EthIf_00007]
EthIfVersionInfoApiMacro	1	[ECUC_EthIf_00008]
EthIfWakeUpSupport	1	[ECUC_EthIf_00040]

Included Containers		
Container Name	**Multiplicity**	**Dependency**
EthIfSecurityEventRefs	0..1	Container for the references to IdsMEvent elements representing the security events that the EthIf module shall report to the IdsM in case the coresponding security related event occurs (and if EthIfEnableSecurityEventReporting is set to "true"). The standardized security events in this container can be extended by vendor-specific security events. **Tags:** atp.Status=draft

⌋

[ECUC_EthIf_00004] Definition of EcucBooleanParamDef EthIfDevErrorDetect ⌈

Parameter Name	EthIfDevErrorDetect		
Parent Container	EthIfGeneral		
Description	Switches the development error detection and notification on or off. • true: detection and notification is enabled. • false: detection and notification is disabled.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency			

⌋

[ECUC_EthIf_00091] Definition of EcucBooleanParamDef EthIfEnableCV2xApi

Status: DRAFT

⌈

Parameter Name	EthIfEnableCV2xApi		
Parent Container	EthIfGeneral		
Description	Enables / Disables API's for CV2x **Tags:** atp.Status=draft		
Multiplicity	0..1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency			

⌋

[ECUC_EthIf_00005] Definition of EcucBooleanParamDef EthIfEnableRxInterrupt ⌈

Parameter Name	EthIfEnableRxInterrupt		
Parent Container	EthIfGeneral		
Description	Enables / Disables receive interrupt.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency			

⌋

[ECUC_EthIf_00079] Definition of EcucBooleanParamDef EthIfEnableSecurityEventReporting

Status: DRAFT

⌈

Parameter Name	EthIfEnableSecurityEventReporting		
Parent Container	EthIfGeneral		
Description	Switches the reporting of security events to the IdsM: - true: reporting is enabled. - false: reporting is disabled. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency			

⌋

[ECUC_EthIf_00076] Definition of EcucBooleanParamDef EthIfEnableSignalQualityApi ⌈

Parameter Name	EthIfEnableSignalQualityApi		
Parent Container	EthIfGeneral		
Description	Enable/disable the APIs read and clear the signal quality.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	

⌋

△

	Post-build time	–	
Dependency			

⌋

[ECUC_EthIf_00006] Definition of EcucBooleanParamDef EthIfEnableTxInterrupt ⌈

Parameter Name	EthIfEnableTxInterrupt		
Parent Container	EthIfGeneral		
Description	Enables / Disables the transmit interrupt.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency			

⌋

[ECUC_EthIf_00075] Definition of EcucBooleanParamDef EthIfEnableWEthApi ⌈

Parameter Name	EthIfEnableWEthApi		
Parent Container	EthIfGeneral		
Description	Enables / Disables API's for WEth / WEthTrcv		
Multiplicity	0..1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency			

⌋

[ECUC_EthIf_00094] Definition of EcucEnumerationParamDef EthIfFwSupport

Status: DRAFT

⌈

Parameter Name	EthIfFwSupport		
Parent Container	EthIfGeneral		
Description	Enables / Disables the Firewall usage. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	FIREWALL_WITHOUT_ PERSTREAMFILTERING	Firewall is used. Network packet is forwarded to Firewall module for inspection. **Tags:** atp.Status=draft	
	FIREWALL_WITH_ PERSTREAMFILTERING	Firewall used with per stream filtering in switch core. Network packet will be forwarded to EthSwt Drv to extract the StreamHandleIdx and afterwards it is forwarded to the Firewall module. **Tags:** atp.Status=draft	
	NO_FIREWALL	No Firewall is used. **Tags:** atp.Status=draft	
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency			

⌋

[ECUC_EthIf_00072] Definition of EcucBooleanParamDef EthIfGetAndResetMeasurementDataApi ⌈

Parameter Name	EthIfGetAndResetMeasurementDataApi		
Parent Container	EthIfGeneral		
Description	Enables / Disables the Get and Reset Measurement Data API		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency			

⌋

[ECUC_EthIf_00034] Definition of EcucBooleanParamDef EthIfGetBaudRate ⌈

Parameter Name	EthIfGetBaudRate
Parent Container	EthIfGeneral
Description	Enables / Disables GetBaudRate API.
Multiplicity	1

▽

△

Type	EcucBooleanParamDef		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency			

⌋

[ECUC_EthIf_00035] Definition of EcucBooleanParamDef EthIfGetCounterState ⌈

Parameter Name	EthIfGetCounterState		
Parent Container	EthIfGeneral		
Description	Enables / Disables GetCounterState API.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency			

⌋

[ECUC_EthIf_00070] Definition of EcucBooleanParamDef EthIfGetCtrlIdxList ⌈

Parameter Name	EthIfGetCtrlIdxList		
Parent Container	EthIfGeneral		
Description	Enables / Disables GetCtrlIdxList API.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency			

⌋

[ECUC_EthIf_00108] Definition of EcucBooleanParamDef EthIfGetPortMacAddrVlanApi ⌈

Parameter Name	EthIfGetPortMacAddrVlanApi
Parent Container	EthIfGeneral
Description	Enables / Disables the GetPortMacAddrVlan API.

▽

△

Multiplicity	0..1		
Type	EcucBooleanParamDef		
Default value	–		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency			

⌋

[ECUC_EthIf_00071] Definition of EcucBooleanParamDef EthIfGetVlanIdSupport ⌈

Parameter Name	EthIfGetVlanIdSupport		
Parent Container	EthIfGeneral		
Description	Enables / Disables GetVlanId API.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency			

⌋

[ECUC_EthIf_00039] Definition of EcucBooleanParamDef EthIfGlobalTimeSupport ⌈

Parameter Name	EthIfGlobalTimeSupport		
Parent Container	EthIfGeneral		
Description	Enables/Disables the Global Time APIs used amongst others by Global Time Synchronization over Ethernet.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency			

⌋

[ECUC_EthIf_00023] Definition of EcucFloatParamDef EthIfMainFunctionPeriod ⌈

Parameter Name	EthIfMainFunctionPeriod		
Parent Container	EthIfGeneral		
Description	Specifies the period of main function EthIf_MainFunctionRx and EthIf_MainFunctionTx in seconds. Ethernet Interface does not require this information but the BSW scheduler.		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	]0 .. INF[		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency			

⌋

[ECUC_EthIf_00056] Definition of EcucFloatParamDef EthIfMainFunctionStatePeriod ⌈

Parameter Name	EthIfMainFunctionStatePeriod		
Parent Container	EthIfGeneral		
Description	Specifies the period of main function EthIf_MainFunctionState in seconds. Ethernet Interface does not require this information but the BSW scheduler.		
Multiplicity	0..1		
Type	EcucFloatParamDef		
Range	]0 .. INF[		
Default value	–		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency	If parameter is defined, then EthIf_MainFunctionState shall be generated.		

⌋

[ECUC_EthIf_00003] Definition of EcucIntegerParamDef EthIfMaxTrcvsTotal ⌈

Parameter Name	EthIfMaxTrcvsTotal	
Parent Container	EthIfGeneral	
Description	Limits the total number of transceivers.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	1 .. 255	

▽

△

Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency			

⌋

[ECUC_EthIf_00102] Definition of EcucBooleanParamDef EthIfPhcSupport

Status: DRAFT

⌈

Parameter Name	EthIfPhcSupport		
Parent Container	EthIfGeneral		
Description	Enables/Disables the PTP HW Clock (PHC). **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency			

⌋

[ECUC_EthIf_00055] Definition of EcucFloatParamDef EthIfPortStartupActiveTime ⌈

Parameter Name	EthIfPortStartupActiveTime		
Parent Container	EthIfGeneral		
Description	Denote the time delay after the mode "ETH_MODE_ACTIVE" of all EthIfSwitchPorts are requested via EthIf_StartAllPorts. This is only used for ports in EthIfSwtPortGroups which are not referenced by any EthIf Controller.		
Multiplicity	0..1		
Type	EcucFloatParamDef		
Range	[0.001 .. 65.535]		
Default value	–		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD

▽

△

	Post-build time	–	
Dependency			

⌋

[ECUC_EthIf_00024] Definition of EcucStringParamDef EthIfPublicCddHeaderFile ⌈

Parameter Name	EthIfPublicCddHeaderFile		
Parent Container	EthIfGeneral		
Description	Defines header files for callback functions which shall be included in case of CDDs. Range of characters is 1.. 32.		
Multiplicity	0..*		
Type	EcucStringParamDef		
Default value	–		
Length	1-32		
Regular Expression	–		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency			

⌋

[ECUC_EthIf_00030] Definition of EcucIntegerParamDef EthIfRxIndicationIterations ⌈

Parameter Name	EthIfRxIndicationIterations		
Parent Container	EthIfGeneral		
Description	Maximum number of Ethernet frames per Ethernet controller polled from the Ethernet driver within EthIf_MainFunctionRx.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 65535		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency			

⌋

[ECUC_EthIf_00062] Definition of EcucBooleanParamDef EthIfSetForwardingModeApi ⌈

Parameter Name	EthIfSetForwardingModeApi		
Parent Container	EthIfGeneral		
Description	Enables /disables EthIf_SetForwardingMode API.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency			

⌋

[ECUC_EthIf_00077] Definition of EcucFloatParamDef EthIfSignalQualityCheckPeriod ⌈

Parameter Name	EthIfSignalQualityCheckPeriod		
Parent Container	EthIfGeneral		
Description	Specifies the period in units of seconds in which the signal quality it polled in the context of EthIf_MainfunctionState. The value shall be an integral multiple of EthIfMain FunctionStatePeriod.		
Multiplicity	0..1		
Type	EcucFloatParamDef		
Range	[-INF .. INF]		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency	If this parameter is defined, the EthIf_MainFunctionState shall be generated and parameter EthIfEnableSignalQualityApi shall be set to TRUE.		

⌋

[ECUC_EthIf_00033] Definition of EcucBooleanParamDef EthIfStartAutoNegotiation ⌈

Parameter Name	EthIfStartAutoNegotiation		
Parent Container	EthIfGeneral		
Description	Enables / Disables StartAutoNegotiation API.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	

▽

△

Dependency	

⌋

[ECUC_EthIf_00064] Definition of EcucBooleanParamDef EthIfSwitchManagementSupport ⌈

Parameter Name	EthIfSwitchManagementSupport		
Parent Container	EthIfGeneral		
Description	Enables/Disables the Switch management APIs to support a Switch-port specific communication attribute access.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency			

⌋

[ECUC_EthIf_00054] Definition of EcucFloatParamDef EthIfSwitchOffPortTimeDelay ⌈

Parameter Name	EthIfSwitchOffPortTimeDelay		
Parent Container	EthIfGeneral		
Description	Denote the time delay after the mode "ETH_MODE_DOWN" of a EthIfSwitchPortGroup will be executed. This is only used for EthIfSwtPortGroups which are not referenced by any EthIf Controller. The time delay shall be greater than the UdpNm timings, because UdpNm shall finish its shutdown handling. (Repeat Message State, Prepare Bus-Sleep state, Bus-Sleep state).		
Multiplicity	0..1		
Type	EcucFloatParamDef		
Range	[0.001 .. 65.535]		
Default value	–		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	–	
Dependency	EthIfSwitchOffPortTimeDelay > (UdpNmTimeoutTime + UdpNmWaitBusSleepTime)		

⌋

[ECUC_EthIf_00009] Definition of EcucIntegerParamDef EthIfTrcvLinkStateChgMainReload ⌈

Parameter Name	EthIfTrcvLinkStateChgMainReload		
Parent Container	EthIfGeneral		
Description	Specifies the frequency of transceiver link state change checks in each period of main function EthIf_MainFunctionTx.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 255		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency			

⌋

[ECUC_EthIf_00063] Definition of EcucBooleanParamDef EthIfVerifyConfigApi ⌈

Parameter Name	EthIfVerifyConfigApi		
Parent Container	EthIfGeneral		
Description	Enables /disables EthIf_VerifyConfig API.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency			

⌋

[ECUC_EthIf_00007] Definition of EcucBooleanParamDef EthIfVersionInfoApi ⌈

Parameter Name	EthIfVersionInfoApi		
Parent Container	EthIfGeneral		
Description	Enables / Disables version info API		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency			

⌋

[ECUC_EthIf_00008] Definition of EcucBooleanParamDef EthIfVersionInfoApi Macro ⌈

Parameter Name	EthIfVersionInfoApiMacro		
Parent Container	EthIfGeneral		
Description	Enables / Disables version info API macro implementation.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency			

⌋

[ECUC_EthIf_00040] Definition of EcucBooleanParamDef EthIfWakeUpSupport ⌈

Parameter Name	EthIfWakeUpSupport		
Parent Container	EthIfGeneral		
Description	Configures if wake-up handling is supported or not: TRUE: wake-up handling is supported FALSE: wake-up handling is not supported This configuration parameter also enables particular other the API at Pre-Compile-Time, e.g. EthIf_CheckWakeup.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency			

⌋

10.2.3 EthIfConfigSet

[ECUC_EthIf_00010] Definition of EcucParamConfContainerDef EthIfConfigSet ⌈

Container Name	EthIfConfigSet
Parent Container	EthIf
Description	Collecting container for all parameters with post-build configuration classes.
Multiplicity	1
Configuration Parameters	

No Included Parameters

Included Containers		
Container Name	**Multiplicity**	**Dependency**
EthIfController	1..*	This container contains the configuration of EthIfController.
EthIfFrameConfig	1..*	Configuration of an Ethernet frame. **Tags:** atp.Status=draft
EthIfPhysController	1..*	This container contains the configuration of EthIfPhysController. The usage of EthIfEthCtrlRef, EthIfCanXLCtrlRef, and EthIfWEth CtrlRef and EthIfCV2xCtrlRef is exclusive OR.
EthIfRxIndicationConfig	1..*	Configuration of receive callback functions.
EthIfSwitch	0..*	This container contains the configuration of EthIfSwitches.
EthIfSwitchPortGroup	0..*	This container contains the configuration of EthIfSwitchPort Groups. If EthIfSwitchPortGroups are controlled by PNC one EthIfSwitch PortGroup per PNC shall exist. The host port shall be part of all EthIfSwitchPortGroups. The up link port of a master switch and the up link port of the slave switch shall be part of all EthIfSwitchPortGroups that contain EthSwtPorts belonging to the slave switch.
EthIfTransceiver	0..*	This container contains the configuration of EthIfTransceiver. The usage of EthIfEthTrcvRef, EthIfCanXLTrcvRef, and EthIf WEthTrcvRef is exclusive OR.
EthIfTrcvLinkStateChgConfig	1..*	Specifies link state change callback function
EthIfTxConfirmationConfig	0..*	Configuration of transmit indication callback functions.

⌋

10.2.4 EthIfController

[ECUC_EthIf_00025] Definition of EcucParamConfContainerDef EthIfController ⌈

Container Name	EthIfController
Parent Container	EthIfConfigSet
Description	This container contains the configuration of EthIfController.
Multiplicity	1..*
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
EthIfCtrlIdx	1	[ECUC_EthIf_00026]
EthIfCtrlMtu	1	[ECUC_EthIf_00032]
EthIfMacSecSupport	0..1	[ECUC_EthIf_00089]
EthIfMaxTxBufsTotal	1	[ECUC_EthIf_00002]
EthIfVlanId	0..1	[ECUC_EthIf_00029]
EthIfEthTrcvRef	0..1	[ECUC_EthIf_00028]
EthIfPaeInstanceRef	0..254	[ECUC_EthIf_00090]
EthIfPhysControllerRef	1	[ECUC_EthIf_00027]
EthIfSwitchRefOrPortGroupRef	0..1	[ECUC_EthIf_00048]

No Included Containers

⌋

[ECUC_EthIf_00026] Definition of EcucIntegerParamDef EthIfCtrlIdx ⌈

Parameter Name	EthIfCtrlIdx		
Parent Container	EthIfController		
Description	This parameter provides a zero-based consecutive index of the Ethernet Communication Controllers. Upper layer BSW modules and the EthIf itself use this index to identify a Ethernet CC.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 255		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency	withAuto = true		

⌋

[ECUC_EthIf_00032] Definition of EcucIntegerParamDef EthIfCtrlMtu ⌈

Parameter Name	EthIfCtrlMtu		
Parent Container	EthIfController		
Description	Specifies the maximum transmission unit (MTU) of the EthIfCtrl in [bytes]. Note: In case a VLAN tag is used for the EthIfCtrl, the frame length of the Ethernet frame will increase by 4 bytes.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	64 .. 9000		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency	1) EthIfVlanId. 2) [Draft] If EthIfController.EthIfMacSecSupport is set to HW_MACSEC or SW_MACSEC then the Mtu will need a proper adaption of the MTU size (MTU size has to be decreased by 24 bytes to avoid packets with a greater size then 1500).		

⌋

[ECUC_EthIf_00089] Definition of EcucEnumerationParamDef EthIfMacSecSupport

Status: DRAFT

⌈

Parameter Name	EthIfMacSecSupport
Parent Container	EthIfController
Description	MACsec support of the ethernet interface controller. **Tags:** atp.Status=draft
Multiplicity	0..1
Type	EcucEnumerationParamDef

▽

△

Range	HW_MACSEC	– **Tags:** atp.Status=draft	
	NO_MACSEC	– **Tags:** atp.Status=draft	
	SW_MACSEC	– **Tags:** atp.Status=draft	
Default value	NO_MACSEC		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

⌋

[ECUC_EthIf_00002] Definition of EcucIntegerParamDef EthIfMaxTxBufsTotal ⌈

Parameter Name	EthIfMaxTxBufsTotal		
Parent Container	EthIfController		
Description	Limits the total number of transmit buffers.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 4294967295		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency			

⌋

[ECUC_EthIf_00029] Definition of EcucIntegerParamDef EthIfVlanId ⌈

Parameter Name	EthIfVlanId		
Parent Container	EthIfController		
Description	A virtual-LAN is identified by this attribute according to IEEE 802.1Q.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 4095		
Default value	–		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME

▽

△

	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

⌋

[ECUC_EthIf_00028] Definition of EcucReferenceDef EthIfEthTrcvRef ⌈

Parameter Name	EthIfEthTrcvRef		
Parent Container	EthIfController		
Description	Reference to an Ethernet transceiver, which is handled by the Ethernet Interface.		
Multiplicity	0..1		
Type	Symbolic name reference to EthIfTransceiver		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

⌋

[ECUC_EthIf_00090] Definition of EcucReferenceDef EthIfPaeInstanceRef

Status: DRAFT

⌈

Parameter Name	EthIfPaeInstanceRef		
Parent Container	EthIfController		
Description	Reference to MkaPaeInstance **Tags:** atp.Status=draft		
Multiplicity	0..254		
Type	Symbolic name reference to MkaPaeInstance		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

⌋

[ECUC_EthIf_00027] Definition of EcucReferenceDef EthIfPhysControllerRef ⌈

Parameter Name	EthIfPhysControllerRef		
Parent Container	EthIfController		
Description	Reference to a physical Ethernet controller, which is handled by the Ethernet Interface.		
Multiplicity	1		
Type	Symbolic name reference to EthIfPhysController		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency	If EthIfEthTrcvRef is configured for an EthIfController then all EthIfController which refer to the same physical controller via EthIfPhysControllerRef (ECUC_EthIf_00027) shall reference the same EthIfTrcv via EthIfEthTrcvRef.		

⌋

[ECUC_EthIf_00048] Definition of EcucChoiceReferenceDef EthIfSwitchRefOrPortGroupRef ⌈

Parameter Name	EthIfSwitchRefOrPortGroupRef		
Parent Container	EthIfController		
Description	The choice reference allows to configure that the EthIfController either references an EthIfSwitch or an EthIfSwitchPortGroup. In case EhIfSwitchPortGroups are controlled by the BswM (e,g, according particular PNC requests), then EthIfSwitchPortGroupRefSemantics shall have the value ETHIF_SWITCH_PORT_GROUP_LINK_INFO. In case EhIfSwitchPortGroups are controlled by the EhtIfController, then EthIfSwitchPortGroupRefSemantics shall have the value ETHIF_SWITCH_PORT_GROUP_CONTROL.		
Multiplicity	0..1		
Type	Choice reference to [EthIfSwitch, EthIfSwitchPortGroup]		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency	* The configuration of EthIfSwitchRefOrPortGroupRef shall only be valid, if this EthIf Controller has no EthIfEthTrcvRef configured. * If EthIfSwitchPortGroups are configured, then all EthIfController which refer to the same EthIfPhysController shall reference an EthIfSwitchPortGroup. * If EthIfSwitchPortGroups are configured, then also EthIfSwitches shall be configured according to the corresponding EthSwtConfig. Those EthIfSwitches shall not be referenced by any EthIfController. (Please note: the EthIfSwitches are used to provide the according EthIfSwitchIdx in the context of EthIf module, which abstracts the underlying switch hardware and is needed in several APIs, e.g. EthSwt_GetSwitchPortWakeupReason).		

⌋

10.2.5 EthIfFrameConfig

[ECUC_EthIf_00109] Definition of EcucParamConfContainerDef EthIfFrameConfig

Status: DRAFT

⌈

Container Name	EthIfFrameConfig		
Parent Container	EthIfConfigSet		
Description	Configuration of an Ethernet frame. **Tags:** atp.Status=draft		
Multiplicity	1..*		
Post-Build Variant Multiplicity	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
EthIfFrameType	1	[ECUC_EthIf_00122]

Included Containers		
Container Name	**Multiplicity**	**Dependency**
EthIfFrameRxPool	0..*	Pool of Rx PDUs for the upper layer modules. Several container instances can be defined for different EthIf Controllers. **Tags:** atp.Status=draft
EthIfFrameTxPool	0..*	Pool of Tx PDUs for the upper layer modules. Several container instances can be defined for different EthIf Controllers or with different EthIfFrameTxPriority. **Tags:** atp.Status=draft

⌋

[ECUC_EthIf_00122] Definition of EcucIntegerParamDef EthIfFrameType

Status: DRAFT

⌈

Parameter Name	EthIfFrameType		
Parent Container	EthIfFrameConfig		
Description	Selects the Ethernet frame type. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 65535		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE

▽

△

	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

⌋

10.2.5.1 EthIfFrameRxPool

[ECUC_EthIf_00116] Definition of EcucParamConfContainerDef EthIfFrameRx Pool

Status: DRAFT

⌈

Container Name	EthIfFrameRxPool		
Parent Container	EthIfFrameConfig		
Description	Pool of Rx PDUs for the upper layer modules. Several container instances can be defined for different EthIfControllers. **Tags:** atp.Status=draft		
Multiplicity	0..*		
Post-Build Variant Multiplicity	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
EthIfFrameRxControllerRef	0..1	[ECUC_EthIf_00120]
EthIfFrameRxPhysControllerRef	0..1	[ECUC_EthIf_00123]

Included Containers		
Container Name	**Multiplicity**	**Dependency**
EthIfFrameRxPdu	0..*	PDU in the EthIfFrameRxPool to be used for the transport of a the corresponding Ethernet frame. Supported MetaDataItemTypes: • TIMETUPLE_TYPE_PTR • ETHERNET_MAC_64 • BROADCAST_8 **Tags:** atp.Status=draft

⌋

[ECUC_EthIf_00120] Definition of EcucReferenceDef EthIfFrameRxControllerRef

Status: DRAFT

⌈

Parameter Name	EthIfFrameRxControllerRef		
Parent Container	EthIfFrameRxPool		
Description	Optional reference to an EthIfController. If this reference is defined then only messages to/from that EthIfController are assigned to the Pdu. With this setup it is possible to define VLAN specific PDUs for dedicated EtherTypes. **Tags:** atp.Status=draft		
Multiplicity	0..1		
Type	Reference to EthIfController		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	true		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency	This reference is mutually exclusive with the reference EthIfFrameRxPhysControllerRef		

⌋

[ECUC_EthIf_00123] Definition of EcucReferenceDef EthIfFrameRxPhysControllerRef ⌈

Parameter Name	EthIfFrameRxPhysControllerRef		
Parent Container	EthIfFrameRxPool		
Description	Optional reference to an EthIfPhysController. If this reference is defined then only messages to/from that EthIfPhysController are assigned to the Pdu. With this setup it is possible to define PDUs for dedicated EtherTypes but for variable VLANs.		
Multiplicity	0..1		
Type	Reference to EthIfPhysController		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	true		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency	This reference is mutually exclusive with the reference EthIfFrameRxControllerRef		

⌋

10.2.5.2 EthIfFrameRxPdu

[ECUC_EthIf_00117] Definition of EcucParamConfContainerDef EthIfFrameRxPdu

Status: DRAFT

⌈

Container Name	EthIfFrameRxPdu		
Parent Container	EthIfFrameRxPool		
Description	PDU in the EthIfFrameRxPool to be used for the transport of a the corresponding Ethernet frame. Supported MetaDataItemTypes: • TIMETUPLE_TYPE_PTR • ETHERNET_MAC_64 • BROADCAST_8 **Tags:** atp.Status=draft		
Multiplicity	0..*		
Post-Build Variant Multiplicity	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
EthIfFrameRxPduId	1	[ECUC_EthIf_00119]
EthIfFrameRxPduRef	1	[ECUC_EthIf_00118]

No Included Containers

⌋

[ECUC_EthIf_00119] Definition of EcucIntegerParamDef EthIfFrameRxPduId

Status: DRAFT

⌈

Parameter Name	EthIfFrameRxPduId		
Parent Container	EthIfFrameRxPdu		
Description	PDU ID used by the upper layer module to indicated completed asynchronous reception of the corresponding Ethernet frame. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 65535		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	

▽

△

	Post-build time	–	
Dependency	withAuto = true		

⌋

[ECUC_EthIf_00118] Definition of EcucReferenceDef EthIfFrameRxPduRef

Status: DRAFT

⌈

Parameter Name	EthIfFrameRxPduRef		
Parent Container	EthIfFrameRxPdu		
Description	Reference to the global PDU representing the Ethernet frame to the upper layer module. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	Reference to Pdu		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

⌋

10.2.5.3 EthIfFrameTxPool

[ECUC_EthIf_00110] Definition of EcucParamConfContainerDef EthIfFrameTxPool

Status: DRAFT

⌈

Container Name	EthIfFrameTxPool		
Parent Container	EthIfFrameConfig		
Description	Pool of Tx PDUs for the upper layer modules. Several container instances can be defined for different EthIfControllers or with different EthIfFrameTxPriority. **Tags:** atp.Status=draft		
Multiplicity	0..*		
Post-Build Variant Multiplicity	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
EthIfFrameTxPriority	0..1	[ECUC_EthIf_00114]
EthIfFrameTxControllerRef	0..1	[ECUC_EthIf_00115]
EthIfFrameTxPhysControllerRef	0..1	[ECUC_EthIf_00124]

Included Containers		
Container Name	**Multiplicity**	**Dependency**
EthIfFrameTxPdu	0..*	PDU in the EthIfFrameTxPool to be used for the transport of the corresponding Ethernet frame. Supported MetaDataItemTypes: • ETHERNET_MAC_64 • TIMETUPLE_TYPE_PTR • LISTELEM_PTR • PRIORITY_8 **Tags:** atp.Status=draft

⌋

[ECUC_EthIf_00114] Definition of EcucIntegerParamDef EthIfFrameTxPriority

Status: DRAFT

⌈

Parameter Name	EthIfFrameTxPriority		
Parent Container	EthIfFrameTxPool		
Description	Definition of a fixed VLAN priority of the pool. This priority shall only be configured if the EthIfController referenced by this EthIfFrame TxPool has a EthIfVlanId. If this parameter is configured, the TxPdus of this pool shall not use MetaDataItems of type PRIORITY_8. **Tags:** atp.Status=draft		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 7		
Default value	–		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency			

⌋

[ECUC_EthIf_00115] Definition of EcucReferenceDef EthIfFrameTxControllerRef

Status: DRAFT

⌈

Parameter Name	EthIfFrameTxControllerRef		
Parent Container	EthIfFrameTxPool		
Description	Optional reference to an EthIfController. If this reference is defined then only messages to/from that EthIfController are assigned to the Pdu. With this setup it is possible to define VLAN specific PDUs for dedicated EtherTypes. **Tags:** atp.Status=draft		
Multiplicity	0..1		
Type	Reference to EthIfController		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	true		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency	This reference is mutually exclusive with the reference EthIfFrameTxPhysControllerRef		

⌋

[ECUC_EthIf_00124] Definition of EcucReferenceDef EthIfFrameTxPhysControllerRef ⌈

Parameter Name	EthIfFrameTxPhysControllerRef		
Parent Container	EthIfFrameTxPool		
Description	Optional reference to an EthIfPhysController. If this reference is defined then only messages to/from that EthIfPhysController are assigned to the Pdu. With this setup it is possible to define PDUs for dedicated EtherTypes but for variable VLANs.		
Multiplicity	0..1		
Type	Reference to EthIfPhysController		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	true		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency	This reference is mutually exclusive with the reference EthIfFrameTxControllerRef		

⌋

10.2.5.4 EthIfFrameTxPdu

[ECUC_EthIf_00111] Definition of EcucParamConfContainerDef EthIfFrameTxPdu

Status: DRAFT

⌈

Container Name	EthIfFrameTxPdu		
Parent Container	EthIfFrameTxPool		
Description	PDU in the EthIfFrameTxPool to be used for the transport of the corresponding Ethernet frame. Supported MetaDataItemTypes: • ETHERNET_MAC_64 • TIMETUPLE_TYPE_PTR • LISTELEM_PTR • PRIORITY_8 **Tags:** atp.Status=draft		
Multiplicity	0..*		
Post-Build Variant Multiplicity	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
EthIfFrameTxPduId	1	[ECUC_EthIf_00113]
EthIfFrameTxPduRef	1	[ECUC_EthIf_00112]

No Included Containers

⌋

[ECUC_EthIf_00113] Definition of EcucIntegerParamDef EthIfFrameTxPduId

Status: DRAFT

⌈

Parameter Name	EthIfFrameTxPduId		
Parent Container	EthIfFrameTxPdu		
Description	PDU ID used by the upper layer module to transmit the corresponding Ethernet frame. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 65535		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	

▽

△

	Post-build time	–	
Dependency	withAuto = true		

⌋

[ECUC_EthIf_00112] Definition of EcucReferenceDef EthIfFrameTxPduRef

Status: DRAFT

⌈

Parameter Name	EthIfFrameTxPduRef		
Parent Container	EthIfFrameTxPdu		
Description	Reference to the global PDU representing the Ethernet frame to the upper layer module. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	Reference to Pdu		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

⌋

10.2.6 EthIfPhysController

[ECUC_EthIf_00045] Definition of EcucParamConfContainerDef EthIfPhysController ⌈

Container Name	EthIfPhysController
Parent Container	EthIfConfigSet
Description	This container contains the configuration of EthIfPhysController. The usage of EthIfEthCtrlRef, EthIfCanXLCtrlRef, and EthIfWEthCtrlRef and EthIfCV2x CtrlRef is exclusive OR.
Multiplicity	1..*
Post-Build Variant Multiplicity	false
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
EthIfPhysControllerIdx	1	[ECUC_EthIf_00046]
EthIfCanXLCtrlRef	0..1	[ECUC_EthIf_00085]
EthIfCV2xCtrlRef	0..1	[ECUC_EthIf_00093]
EthIfEthCtrlRef	0..1	[ECUC_EthIf_00047]
EthIfWEthCtrlRef	0..1	[ECUC_EthIf_00073]

Included Containers		
Container Name	**Multiplicity**	**Dependency**
EthIfClkUnit	0..*	This container contains the configuration of a HW clock unit in an Ethernet Controller. **Tags:** atp.Status=draft
EthIfPhysCtrlRxMainFunction IngressQueueProcessing	0..*	The function checks for new received Ethernet frames at the related Ethernet controller and the related ingress queue referenced via EthIfPhysCtrlRxIngressQueueRef, or at the related CanXL controller and the related ingress FIFO referenced via EthIfCanXLCtrlRxIngressFifoRef. In case of Ethernet controller calling Eth_Receive() with the respective QueueIdx. In case of CanXL controller calling CanXL_Receive() with the respective FifoIdx. **Tags:** atp.Status=draft

⌋

[ECUC_EthIf_00046] Definition of EcucIntegerParamDef EthIfPhysControllerIdx ⌈

Parameter Name	EthIfPhysControllerIdx		
Parent Container	EthIfPhysController		
Description	This parameter provides a zero-based consecutive index of the physical Ethernet controllers. Upper layer BSW modules and the Ethernet Interface itself use this index to identify a physical Ethernet controller.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 255		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency	withAuto = true		

⌋

[ECUC_EthIf_00085] Definition of EcucReferenceDef EthIfCanXLCtrlRef ⌈

Parameter Name	EthIfCanXLCtrlRef		
Parent Container	EthIfPhysController		
Description	Reference to a physical CAN XL controller which is handled by a specific CAN XL driver.		
Multiplicity	0..1		
Type	Symbolic name reference to CanController		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	

▽

△

	Post-build time	–	
Dependency	The referenced CanController has to contain a CanXLController.		

⌋

[ECUC_EthIf_00093] Definition of EcucReferenceDef EthIfCV2xCtrlRef

Status: DRAFT

⌈

Parameter Name	EthIfCV2xCtrlRef		
Parent Container	EthIfPhysController		
Description	Reference to physical Cellular V2X controller, which is handled by a specific Cellular V2 X controller driver **Tags:** atp.Status=draft		
Multiplicity	0..1		
Type	Symbolic name reference to CV2xCtrlConfig		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

⌋

[ECUC_EthIf_00047] Definition of EcucReferenceDef EthIfEthCtrlRef ⌈

Parameter Name	EthIfEthCtrlRef		
Parent Container	EthIfPhysController		
Description	Reference to a physical Ethernet controller, which is handled by a specific Ethernet controller driver.		
Multiplicity	0..1		
Type	Symbolic name reference to EthCtrlConfig		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

⌋

[ECUC_EthIf_00073] Definition of EcucReferenceDef EthIfWEthCtrlRef ⌈

Parameter Name	EthIfWEthCtrlRef		
Parent Container	EthIfPhysController		
Description	Reference to a physical Wireless Ethernet controller, which is handled by a specific Wireless Ethernet controller driver.		
Multiplicity	0..1		
Type	Symbolic name reference to WEthCtrlConfig		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

⌋

10.2.7 EthIfPhysCtrlRxMainFunctionIngressQueueProcessing

[ECUC_EthIf_00106] Definition of EcucParamConfContainerDef EthIfPhysCtrlRx MainFunctionIngressQueueProcessing

Status: DRAFT

⌈

Container Name	EthIfPhysCtrlRxMainFunctionIngressQueueProcessing		
Parent Container	EthIfPhysController		
Description	The function checks for new received Ethernet frames at the related Ethernet controller and the related ingress queue referenced via EthIfPhysCtrlRxIngressQueueRef, or at the related CanXL controller and the related ingress FIFO referenced via EthIfCan XLCtrlRxIngressFifoRef. In case of Ethernet controller calling Eth_Receive() with the respective QueueIdx. In case of CanXL controller calling CanXL_Receive() with the respective FifoIdx. **Tags:** atp.Status=draft		
Multiplicity	0..*		
Post-Build Variant Multiplicity	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
EthIfPhysCtrlRxIndicationIterations	1	[ECUC_EthIf_00052]
EthIfPhysCtrlRxMainFunctionPeriod	1	[ECUC_EthIf_00051]
EthIfCanXLCtrlRxIngressFifoRef	0..1	[ECUC_EthIf_00087]
EthIfPhysCtrlRxIngressQueueRef	0..1	[ECUC_EthIf_00107]

No Included Containers

⌋

[ECUC_EthIf_00052] Definition of EcucIntegerParamDef EthIfPhysCtrlRxIndicationIterations ⌈

Parameter Name	EthIfPhysCtrlRxIndicationIterations		
Parent Container	EthIfPhysCtrlRxMainFunctionIngressQueueProcessing		
Description	Max number of Ethernet frames polled per main function invocation.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 18446744073709551615		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency			

⌋

[ECUC_EthIf_00051] Definition of EcucFloatParamDef EthIfPhysCtrlRxMainFunctionPeriod ⌈

Parameter Name	EthIfPhysCtrlRxMainFunctionPeriod		
Parent Container	EthIfPhysCtrlRxMainFunctionIngressQueueProcessing		
Description	Specifies the period of main function in seconds.		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	[-INF .. INF]		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency			

⌋

[ECUC_EthIf_00087] Definition of EcucReferenceDef EthIfCanXLCtrlRxIngressFifoRef ⌈

Parameter Name	EthIfCanXLCtrlRxIngressFifoRef		
Parent Container	EthIfPhysCtrlRxMainFunctionIngressQueueProcessing		
Description	Reference to the reception FIFO.		
Multiplicity	0..1		
Type	Symbolic name reference to CanXLEthIngressFifo		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	

▽

△

Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency	Mutually exclusive with EthIfPhysCtrlRxIngressQueueRef. One of the two parameters is required.		

⌋

[ECUC_EthIf_00107] Definition of EcucReferenceDef EthIfPhysCtrlRxIngressQueueRef

Status: DRAFT

⌈

Parameter Name	EthIfPhysCtrlRxIngressQueueRef		
Parent Container	EthIfPhysCtrlRxMainFunctionIngressQueueProcessing		
Description	Reference to the ingress Queue. **Tags:** atp.Status=draft		
Multiplicity	0..1		
Type	Symbolic name reference to EthCtrlConfigIngressQueue		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency	Mutually exclusive with EthIfCanXLCtrlRxIngressFifoRef. One of the two parameters is required.		

⌋

10.2.8 EthIfClkUnit

[ECUC_EthIf_00105] Definition of EcucParamConfContainerDef EthIfClkUnit

Status: DRAFT

⌈

Container Name	EthIfClkUnit		
Parent Container	EthIfPhysController		
Description	This container contains the configuration of a HW clock unit in an Ethernet Controller. **Tags:** atp.Status=draft		
Multiplicity	0..*		
Post-Build Variant Multiplicity	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants

▽

△

	Link time	–	
	Post-build time	–	
Configuration Parameters			

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
EthIfClkUnitIdx	1	[ECUC_EthIf_00104]
EthIfClkUnitRef	1	[ECUC_EthIf_00103]

No Included Containers

⌋

[ECUC_EthIf_00104] Definition of EcucIntegerParamDef EthIfClkUnitIdx

Status: DRAFT

⌈

Parameter Name	EthIfClkUnitIdx		
Parent Container	EthIfClkUnit		
Description	Zero-based consecutive index of the HW clock units in the Ethernet Controller. Upper layer BSW modules and the Eth itself use this index to identify a clock in the Ethernet Controller. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 255		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency			

⌋

[ECUC_EthIf_00103] Definition of EcucReferenceDef EthIfClkUnitRef

Status: DRAFT

⌈

Parameter Name	EthIfClkUnitRef		
Parent Container	EthIfClkUnit		
Description	Reference to a HW clock unit which is provided by the Ethernet controller for ingress/ egrees timestamping of frames and optionally to follow PTP time. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	Symbolic name reference to EthClkUnit		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants

▽

△

	Link time	–	
	Post-build time	–	
Dependency			

⌋

10.2.9 EthIfRxIndicationConfig

[ECUC_EthIf_00014] Definition of EcucParamConfContainerDef EthIfRxIndicationConfig ⌈

Container Name	EthIfRxIndicationConfig
Parent Container	EthIfConfigSet
Description	Configuration of receive callback functions.
Multiplicity	1..*
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
EthIfRxIndicationFunction	1	[ECUC_EthIf_00015]

No Included Containers

⌋

[ECUC_EthIf_00015] Definition of EcucFunctionNameDef EthIfRxIndicationFunction ⌈

Parameter Name	EthIfRxIndicationFunction		
Parent Container	EthIfRxIndicationConfig		
Description	Specifies receive indication callback function.		
Multiplicity	1		
Type	EcucFunctionNameDef		
Default value	–		
Regular Expression	–		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

⌋

10.2.10 EthIfSwitch

[ECUC_EthIf_00036] Definition of EcucParamConfContainerDef EthIfSwitch ⌈

Container Name	EthIfSwitch
Parent Container	EthIfConfigSet
Description	This container contains the configuration of EthIfSwitches.
Multiplicity	0..*
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
EthIfSwitchIdx	1	[ECUC_EthIf_00037]
EthIfSwitchRef	1	[ECUC_EthIf_00038]

No Included Containers

⌋

[ECUC_EthIf_00037] Definition of EcucIntegerParamDef EthIfSwitchIdx ⌈

Parameter Name	EthIfSwitchIdx		
Parent Container	EthIfSwitch		
Description	This parameter provides a zero-based consecutive index of the Ethernet Interface Switches. Upper layer BSW modules and the EthIf itself use this index to identify a Ethernet Switch.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 255		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency	withAuto = true		

⌋

[ECUC_EthIf_00038] Definition of EcucReferenceDef EthIfSwitchRef ⌈

Parameter Name	EthIfSwitchRef		
Parent Container	EthIfSwitch		
Description	Reference to a Ethernet Switch, which is handled by a specific Ethernet Switch driver.		
Multiplicity	1		
Type	Symbolic name reference to EthSwtConfig		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

⌋

10.2.11 EthIfSwitchPortGroup

[ECUC_EthIf_00057] Definition of EcucParamConfContainerDef EthIfSwitchPortGroup ⌈

Container Name	EthIfSwitchPortGroup
Parent Container	EthIfConfigSet
Description	This container contains the configuration of EthIfSwitchPortGroups. If EthIfSwitchPortGroups are controlled by PNC one EthIfSwitchPortGroup per PNC shall exist. The host port shall be part of all EthIfSwitchPortGroups. The up link port of a master switch and the up link port of the slave switch shall be part of all EthIfSwitchPortGroups that contain EthSwtPorts belonging to the slave switch.
Multiplicity	0..*
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
EthIfSwitchPortGroupIdx	1	[ECUC_EthIf_00058]
EthIfSwitchPortGroupRefSemantics	0..1	[ECUC_EthIf_00059]
EthIfPortRef	1..*	[ECUC_EthIf_00060]

No Included Containers

⌋

[ECUC_EthIf_00058] Definition of EcucIntegerParamDef EthIfSwitchPortGroupIdx ⌈

Parameter Name	EthIfSwitchPortGroupIdx		
Parent Container	EthIfSwitchPortGroup		
Description	This parameter provides a zero-based consecutive index of the Ethernet Switch Port Groups. Upper layer BSW modules and the EthIf itself use this index to identify an Ethernet Switch Port Group.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 255		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency	withAuto = true		

⌋

[ECUC_EthIf_00059] Definition of EcucEnumerationParamDef EthIfSwitchPortGroupRefSemantics ⌈

Parameter Name	EthIfSwitchPortGroupRefSemantics		
Parent Container	EthIfSwitchPortGroup		
Description	Defines how the EthIfSwitchRefOrPortGroupRef refering to a EthIfSwitchPortGroup shall be interpreted.		
Multiplicity	0..1		
Type	EcucEnumerationParamDef		
Range	ETHIF_SWITCH_PORT_GROUP_CONTROL	Used in case all ports in this group are controlled by the EthIf Controller.	
	ETHIF_SWITCH_PORT_GROUP_LINK_INFO	Used in case all ports in this group are controlled by EthIf_SwitchPortGroupRequestMode.	
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency	only valid if a EthIfSwitchRefOrPortGroupRef refers to the EthIfSwitchPortGroup.		

⌋

[ECUC_EthIf_00060] Definition of EcucReferenceDef EthIfPortRef ⌈

Parameter Name	EthIfPortRef		
Parent Container	EthIfSwitchPortGroup		
Description	Reference to an Ethernet Switch Port.		
Multiplicity	1..*		
Type	Symbolic name reference to EthSwtPort		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

⌋

10.2.12 EthIfTransceiver

[ECUC_EthIf_00042] Definition of EcucParamConfContainerDef EthIfTransceiver ⌈

Container Name	EthIfTransceiver
Parent Container	EthIfConfigSet
Description	This container contains the configuration of EthIfTransceiver. The usage of EthIfEthTrcvRef, EthIfCanXLTrcvRef, and EthIfWEthTrcvRef is exclusive OR.
Multiplicity	0..*
Post-Build Variant Multiplicity	false
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
EthIfQualifiedUnexpectedLinkDownTime	0..1	[ECUC_EthIf_00078]
EthIfTransceiverIdx	1	[ECUC_EthIf_00043]
EthIfCanXLTrcvRef	0..1	[ECUC_EthIf_00086]
EthIfEthTrcvRef	0..1	[ECUC_EthIf_00044]
EthIfWEthTrcvRef	0..1	[ECUC_EthIf_00074]

No Included Containers

⌋

[ECUC_EthIf_00078] Definition of EcucFloatParamDef EthIfQualifiedUnexpectedLinkDownTime ⌈

Parameter Name	EthIfQualifiedUnexpectedLinkDownTime		
Parent Container	EthIfTransceiver		
Description	Specifies the time in seconds an unexpected link down is qualified. This parameter is only used for those Ethernet channels where the ECU act as a passive communication slave (referenced EthTrcv set EthTrcvActAsSlavePassiveEnabled = TRUE). The value shall be a multiple integral of EthIf_MainFunctionState.		
Multiplicity	0..1		
Type	EcucFloatParamDef		
Range	]0 .. INF[		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency	1.) If this parameter is set, EthIf_MainFunctionState has to be available 2.) Only applicable if the referenced EthTrcv has set EthTrcvActAsSlavePassiveEnabled to TRUE.		

⌋

[ECUC_EthIf_00043] Definition of EcucIntegerParamDef EthIfTransceiverIdx ⌈

Parameter Name	EthIfTransceiverIdx		
Parent Container	EthIfTransceiver		
Description	This parameter provides a zero-based consecutive index of the Ethernet transceivers. Upper layer BSW modules and the Ethernet Interface itself use this index to identify an Ethernet tranceiver.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 255		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency	withAuto = true		

⌋

[ECUC_EthIf_00086] Definition of EcucReferenceDef EthIfCanXLTrcvRef ⌈

Parameter Name	EthIfCanXLTrcvRef		
Parent Container	EthIfTransceiver		
Description	Reference to a CAN XL transceiver, which is handled by a specific CAN XL transceiver driver.		
Multiplicity	0..1		
Type	Symbolic name reference to CanTrcvChannel		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency	The referenced CanTrcvChannel has to contain a CanTrcvXLChannel.		

⌋

[ECUC_EthIf_00044] Definition of EcucReferenceDef EthIfEthTrcvRef ⌈

Parameter Name	EthIfEthTrcvRef		
Parent Container	EthIfTransceiver		
Description	Reference to an Ethernet transceiver, which is handled by a specific Ethernet transceiver driver.		
Multiplicity	0..1		
Type	Symbolic name reference to EthTrcvConfig		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

⌋

[ECUC_EthIf_00074] Definition of EcucReferenceDef EthIfWEthTrcvRef ⌈

Parameter Name	EthIfWEthTrcvRef		
Parent Container	EthIfTransceiver		
Description	Reference to an Wireless Ethernet transceiver, which is handled by a specific Wireless Ethernet transceiver driver.		
Multiplicity	0..1		
Type	Symbolic name reference to WEthTrcvConfig		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

⌋

10.2.13 EthIfTrcvLinkStateChgConfig

[ECUC_EthIf_00018] Definition of EcucParamConfContainerDef EthIfTrcvLinkStateChgConfig ⌈

Container Name	EthIfTrcvLinkStateChgConfig
Parent Container	EthIfConfigSet
Description	Specifies link state change callback function
Multiplicity	1..*
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
EthIfTrcvLinkStateChgFunction	1	[ECUC_EthIf_00019]

No Included Containers

⌋

[ECUC_EthIf_00019] Definition of EcucFunctionNameDef EthIfTrcvLinkStateChgFunction ⌈

Parameter Name	EthIfTrcvLinkStateChgFunction		
Parent Container	EthIfTrcvLinkStateChgConfig		
Description	Specifies link state change callback function		
Multiplicity	1		
Type	EcucFunctionNameDef		
Default value	–		
Regular Expression	–		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

⌋

10.2.14 EthIfTxConfirmationConfig

[ECUC_EthIf_00016] Definition of EcucParamConfContainerDef EthIfTxConfirmationConfig ⌈

Container Name	EthIfTxConfirmationConfig
Parent Container	EthIfConfigSet
Description	Configuration of transmit indication callback functions.
Multiplicity	0..*
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
EthIfTxConfirmationFunction	1	[ECUC_EthIf_00017]

No Included Containers

⌋

[ECUC_EthIf_00017] Definition of EcucFunctionNameDef EthIfTxConfirmation Function ⌈

Parameter Name	EthIfTxConfirmationFunction		
Parent Container	EthIfTxConfirmationConfig		
Description	Specifies transmit indication callback function		
Multiplicity	1		
Type	EcucFunctionNameDef		
Default value	–		
Regular Expression	–		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

⌋

10.2.15 EthIfSecurityEventRefs

[ECUC_EthIf_00080] Definition of EcucParamConfContainerDef EthIfSecurity EventRefs

Status: DRAFT

⌈

Container Name	EthIfSecurityEventRefs		
Parent Container	EthIfGeneral		
Description	Container for the references to IdsMEvent elements representing the security events that the EthIf module shall report to the IdsM in case the coresponding security related event occurs (and if EthIfEnableSecurityEventReporting is set to "true"). The standardized security events in this container can be extended by vendor-specific security events. **Tags:** atp.Status=draft		
Multiplicity	0..1		
Post-Build Variant Multiplicity	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
EthIfSEvEthernetFrameMaxLength	0..1	[ECUC_EthIf_00125]
SEV_ETH_DROP_INV_VLAN	0..1	[ECUC_EthIf_00083]
SEV_ETH_DROP_MAC_COLLISION	0..1	[ECUC_EthIf_00084]
SEV_ETH_DROP_UNKNOWN_ETHERTYPE	0..1	[ECUC_EthIf_00081]
SEV_ETH_DROP_VLAN_DOUBLE_TAG	0..1	[ECUC_EthIf_00082]

No Included Containers

⌋

[ECUC_EthIf_00125] Definition of EcucIntegerParamDef EthIfSEvEthernetFrameMaxLength ⌈

Parameter Name	EthIfSEvEthernetFrameMaxLength		
Parent Container	EthIfSecurityEventRefs		
Description	Specifies the maximum length in bytes of the Ethernet Frame contained in the context data of a reported security event.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	1 .. 65535		
Default value	–		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency			

⌋

[ECUC_EthIf_00083] Definition of EcucReferenceDef SEV_ETH_DROP_INV_VLAN

Status: DRAFT

⌈

Parameter Name	SEV_ETH_DROP_INV_VLAN
Parent Container	EthIfSecurityEventRefs
Description	An Ethernet datagram was dropped due to an invalid CrtlIdx/VLAN. **Tags:** atp.Status=draft
Multiplicity	0..1
Type	Symbolic name reference to IdsMEvent
Post-Build Variant Multiplicity	false
Post-Build Variant Value	false

▽

△

Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency			

⌋

[ECUC_EthIf_00084] Definition of EcucReferenceDef SEV_ETH_DROP_MAC_COLLISION

Status: DRAFT

⌈

Parameter Name	SEV_ETH_DROP_MAC_COLLISION		
Parent Container	EthIfSecurityEventRefs		
Description	An Ethernet datagram was dropped because local MAC was same as source MAC in an incoming frame. **Tags:** atp.Status=draft		
Multiplicity	0..1		
Type	Symbolic name reference to IdsMEvent		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency			

⌋

[ECUC_EthIf_00081] Definition of EcucReferenceDef SEV_ETH_DROP_UNKNOWN_ETHERTYPE

Status: DRAFT

⌈

Parameter Name	SEV_ETH_DROP_UNKNOWN_ETHERTYPE
Parent Container	EthIfSecurityEventRefs
Description	An Ethernet datagram was dropped due to an unknown Ethertype. **Tags:** atp.Status=draft
Multiplicity	0..1
Type	Symbolic name reference to IdsMEvent
Post-Build Variant Multiplicity	false
Post-Build Variant Value	false

▽

△

Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency			

⌋

[ECUC_EthIf_00082] Definition of EcucReferenceDef SEV_ETH_DROP_VLAN_DOUBLE_TAG

Status: DRAFT

⌈

Parameter Name	SEV_ETH_DROP_VLAN_DOUBLE_TAG		
Parent Container	EthIfSecurityEventRefs		
Description	An Ethernet datagram was dropped due to double VLAN tag. **Tags:** atp.Status=draft		
Multiplicity	0..1		
Type	Symbolic name reference to IdsMEvent		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency			

⌋

10.3 Configuration constraints

A configuration which deviate from the constraints mentioned in this chapter shall be rejected as invalid.

[SWS_EthIf_CONSTR_00001] ⌈The `EthIfPhysController` and `EthIfTransceiver` shall always refer to the same bus type: If `EthIfPhysController` refers to an `EthIfEthCtrlRef`, `EthIfTransceiver` shall refer to a `EthIfEthTrcvRef`. If `EthIfPhysController` refers to an `EthIfWEthCtrlRef`, `EthIfTransceiver` shall refer to a `EthIfWEthTrcvRef`. If `EthIfPhysController` refers to an `EthIfCanXLCtrlRef`, `EthIfTransceiver` shall refer to a `EthIfCanXLTrcvRef`.⌋

Please note: It is possible to have an EthIfController which do not reference any EthIfTransceiver or EthIfSwitch or EthIfSwitchPortGroup. This is used for a hardware topology where a PHY is controlled by e.g. hardware configuration (e.g. boot strap pins). In this case, the AUTOSAR communication stack would not access the PHY. The PHY will transit to normal mode as soon as the ECU is powered.

[SWS_EthIf_CONSTR_00006] EthIfTransceiver reference configuration constraint for EthIfControllers that refer to the same EthIfPhysController
⌈All EthIfController which refer to the same EthIfPhysController where a <Eth|WEth|CanXL>CtrlDriver is referenced shall reference the same EthIfTransceiver⌋

[SWS_EthIf_CONSTR_00007] EthIfTransceiver reference configuration constraint for an EthIfController that refer to an EthIfPhysController that reference a CanXLCtrlDriver
⌈A EthIfTransceiver which reference a CanXLTransceiverDriver shall be referenced only by a EthIfController which refer to EthIfPhysController that reference a CanXLCtrlDriver⌋

[SWS_EthIf_CONSTR_00008] EthIfTransceiver reference configuration constraint for an EthIfController that refer to an EthIfPhysController that reference an EthCtrlDriver
⌈A EthIfTransceiver which reference a EthTransceiverDriver shall be referenced only by a EthIfController which refer to EthIfPhysController that reference a EthCtrlDriver⌋

[SWS_EthIf_CONSTR_00009] EthIfTransceiver reference configuration constraint for an EthIfController that refer to an EthIfPhysController that reference a WEthCtrlDriver
⌈A EthIfTransceiver which reference a WEthTransceiverDriver shall be referenced only by a EthIfController which refer to EthIfPhysController that reference a WEthCtrlDriver⌋

10.4 Published Information

For details refer to [6] Chapter 10.3 *"Published Information"*.

A Not applicable requirements

[SWS_EthIf_00999]

Upstream requirements: SRS_BSW_00170

⌈These requirements are not applicable to this specification.⌋

B Change History

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

B.1 Change History of this document according to AUTOSAR Release R25-11

B.1.1 Added Constraints in R25-11

Number	Heading
[SWS_EthIf_CONSTR_00011]	Exclude configurations where `EthIfController` reference an `EthIfSwitch` and a `MkaPaeInstance`
[SWS_EthIf_CONSTR_00012]	MACsec configuration consistency per affected `EthIfPhysController`
[SWS_EthIf_CONSTR_00013]	Constraints for MACsec configuration
[SWS_EthIf_CONSTR_00014]	Constraint for MACsec configuration of `EthIfPhysController`
[SWS_EthIf_CONSTR_00015]	Constraint for MACsec configuration regarding a `EthIfController` to process Ethernet frames with EtherType set to the EAPOL Ethernet Type specified by IEEE-802.1X
[SWS_EthIf_CONSTR_00016]	PHY as MACsec protected Ethernet hardware entity
[SWS_EthIf_CONSTR_00017]	Ethernet controller as MACsec protected Ethernet hardware entity
[SWS_EthIf_CONSTR_00018]	Ethernet switch port as MACsec protected Ethernet hardware entity
[SWS_EthIf_CONSTR_00019]	`EthIfController` which reference an `EthIfTransceiver` shall reference at most one `MkaPaeInstance`
[SWS_EthIf_CONSTR_00020]	`EthIfController` which reference an `EthIfTransceiver` shall reference the same `MkaPaeInstance`

▽

△

Number	Heading
[SWS_EthIf_CONSTR_00021]	`EthIfController` which reference an `EthIfPhysController` shall reference the same `MkaPaeInstance`

Table B.1: Added Constraints in R25-11

B.1.2 Changed Constraints in R25-11

none

B.1.3 Deleted Constraints in R25-11

none

B.1.4 Added Specification Items in R25-11

Number	Heading
[ECUC_EthIf_00123]	Definition of EcucReferenceDef EthIfFrameRxPhysControllerRef
[ECUC_EthIf_00124]	Definition of EcucReferenceDef EthIfFrameTxPhysControllerRef
[ECUC_EthIf_00125]	Definition of EcucIntegerParamDef EthIfSEvEthernetFrameMaxLength
[SWS_EthIf_00201]	Forward a call of EthIf_GetRxMgmtObject via EthSwt_GetRxMgmtObject to EthSwt
[SWS_EthIf_00202]	Forward a call of EthIf_GetTxMgmtObject via EthSwt_GetTxMgmtObject to EthSwt
[SWS_EthIf_00203]	Forward a call of EthIf_SetSwitchMgmtInfo via EthSwt_SetMgmtInfo to Eth Swt
[SWS_EthIf_00204]	Forward a call of EthIf_SwitchEnableTimeStamping via EthSwt_PortEnable TimeStamp to EthSwt
[SWS_EthIf_00679]	DET error reporting of `ETHIF_E_INV_CLKUNIT_IDX`
[SWS_EthIf_00680]	Supervise and store the link state per configured Ethernet hardware entity
[SWS_EthIf_00681]	Handling unavailable forwarding APIs
[SWS_EthIf_00682]	Report of a link state change towards the EthSM and configured users for an `EthIfController` that acts independent of a MACsec operational state
[SWS_EthIf_00683]	Always use the sole link state of a MACsec protected Ethernet hardware entities to report a link state change towards Mka
[SWS_EthIf_00684]	Report of an detected link state change for MACsec protected Ethernet hardware entities towards Mka

▽

△

Number	Heading
[SWS_EthIf_00685]	Report of a link state change towards the EthSM and configured users for an EthIfController that depends on a MACsec operational state and references an EthIfTransceiver
[SWS_EthIf_00686]	Report of a link state change towards the EthSM and configured users for an EthIfController that references an EthIfSwitchPortGroup
[SWS_EthIf_00687]	Derivation of the EthSwtPort link state used for the link state accumulation
[SWS_EthIf_00688]	Indication of EthIf_MacSecOperational with a CtrlIdx that refers to an EthIfController that is referenced by a MkaPaeInstance
[SWS_EthIf_00689]	Indication of EthIf_SwitchMacSecOperational with a MgmtInfoPtr where the given EthSwtPort is referenced by a MkaPaeInstance
[SWS_EthIf_00690]	Forwarding of MACsec related API calls towards the Eth driver
[SWS_EthIf_00691]	Forwarding of MACsec related API calls towards the EthTrcv driver
[SWS_EthIf_00692]	Forwarding of MACsec related API calls towards the EthTrcv driver where a EthTrcvConfig is referenced by an EthSwtPort
[SWS_EthIf_00693]	Forwarding of MACsec related API calls towards the EthSwt driver where the EthSwtPort has MACsec support enabled
[SWS_EthIf_00694]	Forwarding of MACsec related callback APIs calls towards the MKA module called by the Eth driver
[SWS_EthIf_00695]	Forwarding of MACsec related callback APIs calls towards the MKA module called by the EthTrcv driver
[SWS_EthIf_00696]	Forwarding of MACsec related callback APIs calls towards the MKA module called by the EthTrcv driver where the EthTrcvConfig is referenced by an EthSwtPort
[SWS_EthIf_00697]	Either use sole link state or accumulated to indicate a link state change towards EthSM, BswM and configured users
[SWS_EthIf_00698]	Trigger condition of SEV_ETH_DROP_UNKNOWN_ETHERTYPE
[SWS_EthIf_00699]	Security event context data definition: SEV_ETH_DROP_UNKNOWN_ETHERTYPE
[SWS_EthIf_00700]	Trigger condition of SEV_ETH_DROP_VLAN_DOUBLE_TAG
[SWS_EthIf_00701]	Security event context data definition: SEV_ETH_DROP_VLAN_DOUBLE_TAG
[SWS_EthIf_00702]	Trigger condition of SEV_ETH_DROP_INV_VLAN
[SWS_EthIf_00703]	Security event context data definition: SEV_ETH_DROP_INV_VLAN
[SWS_EthIf_00704]	Trigger condition of SEV_ETH_DROP_MAC_COLLISION
[SWS_EthIf_00705]	Security event context data definition: SEV_ETH_DROP_MAC_COLLISION
[SWS_EthIf_00706]	Forwarding a call of EthIf_ReadMii to Eth_ReadMii
[SWS_EthIf_00707]	Forwarding a call of EthIf_WriteMii to Eth_WriteMii
[SWS_EthIf_00708]	Forwarding a call of EthIf_ReadMmd to Eth_ReadMmd
[SWS_EthIf_00709]	Forwarding a call of EthIf_WriteMmd to Eth_WriteMmd
[SWS_EthIf_00710]	Forwarding a call of EthIf_SwitchPortReadTrcvRegister to EthSwt_ReadTrcvRegister
[SWS_EthIf_00711]	Forwarding a call of EthIf_SwitchPortWriteTrcvRegiste to EthSwt_WriteTrcvRegister

▽

△

Number	Heading
[SWS_EthIf_00712]	Forwarding a call of EthIf_SwitchPortReadMmd to EthSwt_ReadMmd
[SWS_EthIf_00713]	Forwarding a call of EthIf_SwitchPortWriteMmd to EthSwt_WriteMmd
[SWS_EthIf_00714]	Forwarding a call of EthIf_SetTransceiverMode to EthTrcv_SetTransceiver Mode
[SWS_EthIf_00715]	Forwarding a call of EthIf_StartAutoNegotiation to EthTrcv_StartAuto Negotiation
[SWS_EthIf_91239]	Definition of API function EthIf_ReadMii
[SWS_EthIf_91240]	Definition of API function EthIf_WriteMii
[SWS_EthIf_91241]	Definition of API function EthIf_ReadMmd
[SWS_EthIf_91242]	Definition of API function EthIf_WriteMmd
[SWS_EthIf_91243]	Definition of API function EthIf_SwtEthRxProcessFrame
[SWS_EthIf_91244]	Definition of API function EthIf_SwtEthRxFinishedIndication
[SWS_EthIf_91245]	Definition of API function EthIf_SwtEthTxPrepareFrame
[SWS_EthIf_91246]	Definition of API function EthIf_SwtEthTxAdaptBufferLength
[SWS_EthIf_91247]	Definition of API function EthIf_SwtEthTxProcessFrame
[SWS_EthIf_91248]	Definition of API function EthIf_SwtEthTxFinishedIndication
[SWS_EthIf_91249]	Definition of API function EthIf_SwitchPortReadTrcvRegister
[SWS_EthIf_91250]	Definition of API function EthIf_SwitchPortWriteTrcvRegister
[SWS_EthIf_91251]	Definition of API function EthIf_SwitchPortReadMmd
[SWS_EthIf_91252]	Definition of API function EthIf_SwitchPortWriteMmd
[SWS_EthIf_91253]	Definition of callback function EthIf_EthMacSecUpdateSecYNotification
[SWS_EthIf_91254]	Definition of callback function EthIf_EthMacSecAddRxSaNotification
[SWS_EthIf_91255]	Definition of callback function EthIf_EthMacSecAddTxSaNotification
[SWS_EthIf_91256]	Definition of callback function EthIf_EthMacSecGetMacSecStatistics Notification
[SWS_EthIf_91257]	Definition of API function EthIf_SetTransceiverMode
[SWS_EthIf_91258]	Definition of API function EthIf_StartAutoNegotiation

Table B.2: Added Specification Items in R25-11

B.1.5 Changed Specification Items in R25-11

Number	Heading
[ECUC_EthIf_00080]	Definition of EcucParamConfContainerDef EthIfSecurityEventRefs
[ECUC_EthIf_00106]	Definition of EcucParamConfContainerDef EthIfPhysCtrlRxMainFunction IngressQueueProcessing
[ECUC_EthIf_00110]	Definition of EcucParamConfContainerDef EthIfFrameTxPool
[ECUC_EthIf_00113]	Definition of EcucIntegerParamDef EthIfFrameTxPduId

▽

△

Number	Heading
[ECUC_EthIf_00115]	Definition of EcucReferenceDef EthIfFrameTxControllerRef
[ECUC_EthIf_00116]	Definition of EcucParamConfContainerDef EthIfFrameRxPool
[ECUC_EthIf_00119]	Definition of EcucIntegerParamDef EthIfFrameRxPduId
[ECUC_EthIf_00120]	Definition of EcucReferenceDef EthIfFrameRxControllerRef
[SWS_EthIf_00017]	Definition of development errors in module EthIf
[SWS_EthIf_00023]	Definition of imported datatypes of module EthIf
[SWS_EthIf_00103]	Definition of optional interfaces requested by module EthIf
[SWS_EthIf_00132]	Definition of API function EthIf_SetPhysAddr
[SWS_EthIf_00139]	Definition of API function EthIf_UpdatePhysAddrFilter
[SWS_EthIf_00140]	
[SWS_EthIf_00166]	Definition of API function EthIf_GetEgressTimeStamp
[SWS_EthIf_00172]	Definition of API function EthIf_GetIngressTimeStamp
[SWS_EthIf_00176]	
[SWS_EthIf_00259]	Link state accumulation of an `EthIfSwitchPortGroup`
[SWS_EthIf_00324]	
[SWS_EthIf_00327]	
[SWS_EthIf_00330]	
[SWS_EthIf_00334]	
[SWS_EthIf_00388]	
[SWS_EthIf_00391]	
[SWS_EthIf_00417]	
[SWS_EthIf_00421]	
[SWS_EthIf_00426]	
[SWS_EthIf_00430]	
[SWS_EthIf_00466]	
[SWS_EthIf_00474]	
[SWS_EthIf_00486]	
[SWS_EthIf_00505]	
[SWS_EthIf_00600]	Finalization of transmission request with direct or indirect data provision
[SWS_EthIf_00638]	Error handling for aborted reception indication process
[SWS_EthIf_00639]	Recpetion handling for PDUs with `KeepLocalPduBuffer` set to `FALSE`
[SWS_EthIf_00641]	Handling if `EthIf_ReleaseRxBuffer` is called
[SWS_EthIf_00644]	
[SWS_EthIf_00647]	
[SWS_EthIf_00666]	Transmission request with direct data provision and immediate forwarding
[SWS_EthIf_00667]	Creation of a list-element-struct of type `ListElemStructType`
[SWS_EthIf_00671]	Transmission request with direct data provision and deferred forwarding
[SWS_EthIf_00672]	Preparation for call `<EthDrv>_ProvideTxBuffer`

▽

△

Number	Heading
[SWS_EthIf_00673]	Return of `<EthDrv>_ProvideTxBuffer` for transmission request with indirect data provision
[SWS_EthIf_00674]	Return of `LSduR_EthIfTriggerTransmit` for transmission request with indirect data provision
[SWS_EthIf_00675]	`<EthDrv>_Transmit` return `E_NOT_OK` for transmission request with indirect data provision
[SWS_EthIf_00676]	Transmission request with indirect data provision
[SWS_EthIf_00677]	Return of `<EthDrv>_ProvideTxBuffer` for transmission request with direct data provision and deferred forwarding
[SWS_EthIf_00678]	`<EthDrv>_Transmit` return `E_NOT_OK` for transmission request with direct data provision and deferred forwarding
[SWS_EthIf_91021]	Definition of API function EthIf_TransceiverGetMacMethod
[SWS_EthIf_91022]	Definition of API function EthIf_EthGetSpiStatus
[SWS_EthIf_91063]	Definition of API function EthIf_SetPhcCorrection
[SWS_EthIf_91108]	Definition of API function EthIf_GetTransceiverMode
[SWS_EthIf_91110]	Definition of API function EthIf_TransceiverGetLinkState
[SWS_EthIf_91112]	Definition of API function EthIf_TransceiverGetBaudRate
[SWS_EthIf_91114]	Definition of API function EthIf_TransceiverGetDuplexMode
[SWS_EthIf_91132]	Definition of API function EthIf_RunCableDiagnostic

Table B.3: Changed Specification Items in R25-11

B.1.6 Deleted Specification Items in R25-11

Number	Heading
[ECUC_EthIf_00053]	Definition of EcucReferenceDef EthIfPhysCtrlRxIngressFifoRef
[SWS_EthIf_00008]	
[SWS_EthIf_00154]	Definition of API function EthIf_GetCurrentTime
[SWS_EthIf_00155]	
[SWS_EthIf_00157]	
[SWS_EthIf_00158]	
[SWS_EthIf_00260]	
[SWS_EthIf_00262]	
[SWS_EthIf_00319]	
[SWS_EthIf_00473]	
[SWS_EthIf_00581]	
[SWS_EthIf_00582]	
[SWS_EthIf_00603]	
[SWS_EthIf_00609]	

▽

△

Number	Heading
[SWS_EthIf_00622]	
[SWS_EthIf_00627]	
[SWS_EthIf_91051]	Definition of scheduled function EthIf_MainFunctionRx_<PriorityProcessing ShortName>

Table B.4: Deleted Specification Items in R25-11

B.2 Change History of this document according to AUTOSAR Release R24-11

B.2.1 Added Constraints in R24-11

Number	Heading
[SWS_EthIf_CONSTR_00006]	`EthIfTransceiver` reference configuration constraint for `EthIfController`s that refer to the same `EthIfPhysController`
[SWS_EthIf_CONSTR_00007]	`EthIfTransceiver` reference configuration constraint for an EthIfController that refer to an `EthIfPhysController` that reference a `CanXLCtrlDriver`
[SWS_EthIf_CONSTR_00008]	`EthIfTransceiver` reference configuration constraint for an `EthIfController` that refer to an `EthIfPhysController` that reference an `EthCtrlDriver`
[SWS_EthIf_CONSTR_00009]	`EthIfTransceiver` reference configuration constraint for an `EthIfController` that refer to an `EthIfPhysController` that reference a `WEthCtrlDriver`
[SWS_EthIf_CONSTR_00010]	Same configuration of PDUs that belong to the same PDU pool for `KeepLocalPduBuffer`

Table B.5: Added Constraints in R24-11

B.2.2 Changed Constraints in R24-11

Number	Heading
[SWS_EthIf_CONSTR_00002]	
[SWS_EthIf_CONSTR_00003]	

▽

△

Number	Heading
[SWS_EthIf_CONSTR_00004]	
[SWS_EthIf_CONSTR_00005]	

Table B.6: Changed Constraints in R24-11

B.2.3 Deleted Constraints in R24-11

none

B.2.4 Added Specification Items in R24-11

Number	Heading
[ECUC_EthIf_00108]	Definition of EcucBooleanParamDef EthIfGetPortMacAddrVlanApi
[ECUC_EthIf_00109]	Definition of EcucParamConfContainerDef EthIfFrameConfig
[ECUC_EthIf_00110]	Definition of EcucParamConfContainerDef EthIfFrameTxPool
[ECUC_EthIf_00111]	Definition of EcucParamConfContainerDef EthIfFrameTxPdu
[ECUC_EthIf_00112]	Definition of EcucReferenceDef EthIfFrameTxPduRef
[ECUC_EthIf_00113]	Definition of EcucIntegerParamDef EthIfFrameTxPduId
[ECUC_EthIf_00114]	Definition of EcucIntegerParamDef EthIfFrameTxPriority
[ECUC_EthIf_00115]	Definition of EcucReferenceDef EthIfFrameTxControllerRef
[ECUC_EthIf_00116]	Definition of EcucParamConfContainerDef EthIfFrameRxPool
[ECUC_EthIf_00117]	Definition of EcucParamConfContainerDef EthIfFrameRxPdu
[ECUC_EthIf_00118]	Definition of EcucReferenceDef EthIfFrameRxPduRef
[ECUC_EthIf_00119]	Definition of EcucIntegerParamDef EthIfFrameRxPduId
[ECUC_EthIf_00120]	Definition of EcucReferenceDef EthIfFrameRxControllerRef
[ECUC_EthIf_00122]	Definition of EcucIntegerParamDef EthIfFrameType
[SWS_EthIf_00649]	Controller mode request `ETH_MODE_ACTIVE` or `ETH_MODE_ACTIVE_WITH_WAKEUP_REQUEST` for an EthIf controller which is not referencing a transceiver, switch or switch port group
[SWS_EthIf_00650]	Controller mode request `ETH_MODE_DOWN` for an EthIf controller which is not referencing a transceiver, switch or switch port group
[SWS_EthIf_00651]	DET error reporting of `ETHIF_E_UNINIT`
[SWS_EthIf_00652]	DET error reporting of `ETHIF_E_PARAM_POINTER`
[SWS_EthIf_00653]	DET error reporting of `ETHIF_E_INV_CTRL_IDX`

▽

△

Number	Heading
[SWS_EthIf_00654]	DET error reporting of ETHIF_E_INV_TRCV_IDX
[SWS_EthIf_00655]	DET error reporting of ETHIF_E_INV_SWT_IDX
[SWS_EthIf_00656]	DET error reporting of ETHIF_E_INV_PORT_GROUP_IDX
[SWS_EthIf_00657]	DET error reporting of ETHIF_E_INV_PORT_IDX
[SWS_EthIf_00658]	DET error reporting of ETHIF_E_INV_PARAM
[SWS_EthIf_00659]	Behaviour if function is called
[SWS_EthIf_00660]	Pre compile configuration switch
[SWS_EthIf_00661]	Setting of TrcvLinkState in configured state change function when the referenced controller is not referencing a transceiver, nor a switch or switch port group
[SWS_EthIf_00662]	Forwarding of stream statistics indications to firewall module
[SWS_EthIf_00663]	Reception handling with fixed EthIfFrameRxPools
[SWS_EthIf_00664]	Reception handling with floating EthIfFrameRxPools
[SWS_EthIf_00665]	Abort of reception indication process
[SWS_EthIf_00666]	Transmission request with direct data provision and immediate forwarding
[SWS_EthIf_00667]	Creation of a list-element-struct of type ListElemStructType
[SWS_EthIf_00668]	Handling if EthIfFwSupport is set to FIREWALL_WITHOUT_PERSTREAMFILTERING
[SWS_EthIf_00669]	Handling if EthIfFwSupport is set to FIREWALL_WITH_PERSTREAMFILTERING
[SWS_EthIf_00670]	Evaluation of transmission request with direct data provision
[SWS_EthIf_00671]	Transmission request with direct data provision and deferred forwarding
[SWS_EthIf_00672]	Preparation for call Eth_ProvideTxBuffer
[SWS_EthIf_00673]	Return of Eth_ProvideTxBuffer for transmission request with indirect data provision
[SWS_EthIf_00674]	Return of LSduR_EthIfTriggerTransmit for transmission request with indirect data provision
[SWS_EthIf_00675]	Eth_Transmit return E_NOT_OK for transmission request with indirect data provision
[SWS_EthIf_00676]	Transmission request with indirect data provision
[SWS_EthIf_00677]	Return of Eth_ProvideTxBuffer for transmission request with direct data provision and deferred forwarding
[SWS_EthIf_00678]	Eth_Transmit return E_NOT_OK for transmission request with direct data provision and deferred forwarding
[SWS_EthIf_91140]	Definition of API function EthIf_GetPortMacAddrVlan

Table B.7: Added Specification Items in R24-11

B.2.5 Changed Specification Items in R24-11

Number	Heading
[ECUC_EthIf_00001]	Definition of EcucParamConfContainerDef EthIfGeneral
[ECUC_EthIf_00010]	Definition of EcucParamConfContainerDef EthIfConfigSet
[ECUC_EthIf_00025]	Definition of EcucParamConfContainerDef EthIfController
[ECUC_EthIf_00090]	Definition of EcucReferenceDef EthIfPaeInstanceRef
[SWS_EthIf_00023]	Definition of imported datatypes of module EthIf
[SWS_EthIf_00075]	Definition of API function EthIf_Transmit
[SWS_EthIf_00160]	Definition of API function EthIf_EnableEgressTimeStamp
[SWS_EthIf_00166]	Definition of API function EthIf_GetEgressTimeStamp
[SWS_EthIf_00172]	Definition of API function EthIf_GetIngressTimeStamp
[SWS_EthIf_00192]	
[SWS_EthIf_00472]	
[SWS_EthIf_00503]	Security events for EthIf
[SWS_EthIf_00592]	
[SWS_EthIf_00593]	
[SWS_EthIf_00600]	Finalization of transmission request with direct or indirect data provision
[SWS_EthIf_00639]	Recpetion handling for PDUs with `KeepLocalPduBuffer` set to `FALSE`
[SWS_EthIf_00641]	Handling if `EthIf_ReleaseRxBuffer` is called
[SWS_EthIf_00644]	
[SWS_EthIf_00647]	
[SWS_EthIf_91003]	Definition of API function EthIf_SetSwitchMgmtInfo
[SWS_EthIf_91007]	Definition of API function EthIf_SwitchEnableTimeStamping
[SWS_EthIf_91017]	Definition of API function EthIf_SetBufWTxParams
[SWS_EthIf_91023]	Definition of callback function EthIf_StreamStatisticsIndication
[SWS_EthIf_91024]	Definition of callback function EthIf_StreamStateIndication
[SWS_EthIf_91025]	Definition of API function EthIf_SetStreamState
[SWS_EthIf_91027]	Definition of API function EthIf_GetStreamStatistics
[SWS_EthIf_91105]	Definition of API function EthIf_GetRxMgmtObject
[SWS_EthIf_91106]	Definition of API function EthIf_GetTxMgmtObject
[SWS_EthIf_91135]	Definition of API function EthIf_SwitchPortGetMaxQueueBufferFillLevel
[SWS_EthIf_91201]	Definition of API function EthIf_GetBufCV2xPC5RxParams
[SWS_EthIf_91202]	Definition of API function EthIf_GetBufCV2xPC5TxParams
[SWS_EthIf_91203]	Definition of API function EthIf_SetBufCV2xPC5TxParams
[SWS_EthIf_91209]	Definition of API function EthIf_MacSecGetMacSecStatistics
[SWS_EthIf_91212]	Definition of API function EthIf_MacSecOperational
[SWS_EthIf_91217]	Definition of callback function EthIf_SwitchMacSecUpdateSecYNotification
[SWS_EthIf_91218]	Definition of callback function EthIf_MacSecUpdateSecYNotification

▽

△

Number	Heading
[SWS_EthIf_91223]	Definition of callback function EthIf_SwitchMacSecAddTxSaNotification
[SWS_EthIf_91224]	Definition of callback function EthIf_MacSecAddTxSaNotification
[SWS_EthIf_91228]	Definition of callback function EthIf_SwitchMacSecAddRxSaNotification
[SWS_EthIf_91229]	Definition of callback function EthIf_MacSecAddRxSaNotification
[SWS_EthIf_91233]	Definition of API function EthIf_SwitchMacSecGetMacSecStatistics
[SWS_EthIf_91234]	Definition of callback function EthIf_SwitchMacSecGetMacSecStatistics Notification
[SWS_EthIf_91235]	Definition of callback function EthIf_MacSecGetMacSecStatisticsNotification
[SWS_EthIf_91236]	Definition of API function EthIf_SwitchMacSecOperational

Table B.8: Changed Specification Items in R24-11

B.2.6 Deleted Specification Items in R24-11

Number	Heading
[ECUC_EthIf_00011]	Definition of EcucParamConfContainerDef EthIfFrameOwnerConfig
[ECUC_EthIf_00012]	Definition of EcucIntegerParamDef EthIfFrameType
[ECUC_EthIf_00013]	Definition of EcucIntegerParamDef EthIfOwner
[ECUC_EthIf_00095]	Definition of EcucParamConfContainerDef EthIfFrameOwnerPdu
[ECUC_EthIf_00096]	Definition of EcucParamConfContainerDef EthIfFrameOwnerPduPoolEntry
[ECUC_EthIf_00097]	Definition of EcucReferenceDef EthIfFrameOwnerPduRef
[ECUC_EthIf_00098]	Definition of EcucIntegerParamDef EthIfFrameOwnerPduId
[ECUC_EthIf_00099]	Definition of EcucEnumerationParamDef EthIfFrameOwnerPduDirection
[ECUC_EthIf_00100]	Definition of EcucIntegerParamDef EthIfFrameOwnerTxPriority
[ECUC_EthIf_00101]	Definition of EcucReferenceDef EthIfFrameOwnerControllerRef
[SWS_EthIf_00036]	
[SWS_EthIf_00037]	
[SWS_EthIf_00041]	
[SWS_EthIf_00042]	
[SWS_EthIf_00043]	
[SWS_EthIf_00063]	
[SWS_EthIf_00064]	
[SWS_EthIf_00065]	
[SWS_EthIf_00067]	Definition of API function EthIf_ProvideTxBuffer
[SWS_EthIf_00068]	
[SWS_EthIf_00069]	
[SWS_EthIf_00070]	
[SWS_EthIf_00071]	

▽

△

Number	Heading
[SWS_EthIf_00072]	
[SWS_EthIf_00073]	
[SWS_EthIf_00077]	
[SWS_EthIf_00078]	
[SWS_EthIf_00079]	
[SWS_EthIf_00080]	
[SWS_EthIf_00086]	
[SWS_EthIf_00087]	
[SWS_EthIf_00088]	
[SWS_EthIf_00092]	
[SWS_EthIf_00093]	
[SWS_EthIf_00094]	
[SWS_EthIf_00104]	Definition of configurable interface <User>_RxIndication
[SWS_EthIf_00105]	
[SWS_EthIf_00106]	Definition of configurable interface <UL>_TxConfirmation
[SWS_EthIf_00107]	
[SWS_EthIf_00125]	
[SWS_EthIf_00127]	
[SWS_EthIf_00135]	
[SWS_EthIf_00136]	
[SWS_EthIf_00137]	
[SWS_EthIf_00141]	
[SWS_EthIf_00142]	
[SWS_EthIf_00143]	
[SWS_EthIf_00146]	
[SWS_EthIf_00147]	
[SWS_EthIf_00156]	
[SWS_EthIf_00161]	
[SWS_EthIf_00162]	
[SWS_EthIf_00167]	
[SWS_EthIf_00168]	
[SWS_EthIf_00169]	
[SWS_EthIf_00173]	
[SWS_EthIf_00174]	
[SWS_EthIf_00175]	
[SWS_EthIf_00193]	
[SWS_EthIf_00194]	
[SWS_EthIf_00199]	

▽

△

Number	Heading
[SWS_EthIf_00200]	
[SWS_EthIf_00217]	
[SWS_EthIf_00222]	
[SWS_EthIf_00229]	
[SWS_EthIf_00246]	
[SWS_EthIf_00247]	
[SWS_EthIf_00273]	
[SWS_EthIf_00274]	
[SWS_EthIf_00277]	
[SWS_EthIf_00280]	
[SWS_EthIf_00281]	
[SWS_EthIf_00282]	
[SWS_EthIf_00283]	
[SWS_EthIf_00286]	
[SWS_EthIf_00287]	
[SWS_EthIf_00288]	
[SWS_EthIf_00289]	
[SWS_EthIf_00290]	
[SWS_EthIf_00299]	
[SWS_EthIf_00300]	
[SWS_EthIf_00302]	
[SWS_EthIf_00303]	
[SWS_EthIf_00304]	
[SWS_EthIf_00306]	
[SWS_EthIf_00325]	
[SWS_EthIf_00326]	
[SWS_EthIf_00328]	
[SWS_EthIf_00329]	
[SWS_EthIf_00331]	
[SWS_EthIf_00332]	
[SWS_EthIf_00333]	
[SWS_EthIf_00335]	
[SWS_EthIf_00336]	
[SWS_EthIf_00337]	
[SWS_EthIf_00338]	
[SWS_EthIf_00339]	
[SWS_EthIf_00343]	
[SWS_EthIf_00344]	

▽

△

Number	Heading
[SWS_EthIf_00345]	
[SWS_EthIf_00346]	
[SWS_EthIf_00349]	
[SWS_EthIf_00350]	
[SWS_EthIf_00351]	
[SWS_EthIf_00352]	
[SWS_EthIf_00355]	
[SWS_EthIf_00356]	
[SWS_EthIf_00357]	
[SWS_EthIf_00358]	
[SWS_EthIf_00359]	
[SWS_EthIf_00362]	
[SWS_EthIf_00363]	
[SWS_EthIf_00364]	
[SWS_EthIf_00365]	
[SWS_EthIf_00368]	
[SWS_EthIf_00369]	
[SWS_EthIf_00370]	
[SWS_EthIf_00371]	
[SWS_EthIf_00372]	
[SWS_EthIf_00375]	
[SWS_EthIf_00376]	
[SWS_EthIf_00377]	
[SWS_EthIf_00378]	
[SWS_EthIf_00379]	
[SWS_EthIf_00382]	
[SWS_EthIf_00383]	
[SWS_EthIf_00384]	
[SWS_EthIf_00385]	
[SWS_EthIf_00386]	
[SWS_EthIf_00389]	
[SWS_EthIf_00390]	
[SWS_EthIf_00392]	
[SWS_EthIf_00393]	
[SWS_EthIf_00394]	
[SWS_EthIf_00396]	
[SWS_EthIf_00397]	
[SWS_EthIf_00399]	

▽

△

Number	Heading
[SWS_EthIf_00401]	
[SWS_EthIf_00402]	
[SWS_EthIf_00405]	
[SWS_EthIf_00406]	
[SWS_EthIf_00475]	
[SWS_EthIf_00476]	
[SWS_EthIf_00477]	
[SWS_EthIf_00487]	
[SWS_EthIf_00488]	
[SWS_EthIf_00489]	
[SWS_EthIf_00491]	
[SWS_EthIf_00492]	
[SWS_EthIf_00493]	
[SWS_EthIf_00494]	
[SWS_EthIf_00495]	
[SWS_EthIf_00496]	
[SWS_EthIf_00506]	
[SWS_EthIf_00507]	
[SWS_EthIf_00508]	
[SWS_EthIf_00523]	
[SWS_EthIf_00524]	
[SWS_EthIf_00525]	
[SWS_EthIf_00526]	
[SWS_EthIf_00533]	
[SWS_EthIf_00534]	
[SWS_EthIf_00535]	
[SWS_EthIf_00536]	
[SWS_EthIf_00543]	
[SWS_EthIf_00544]	
[SWS_EthIf_00545]	
[SWS_EthIf_00546]	
[SWS_EthIf_00547]	
[SWS_EthIf_00553]	
[SWS_EthIf_00554]	
[SWS_EthIf_00555]	
[SWS_EthIf_00556]	
[SWS_EthIf_00557]	
[SWS_EthIf_00587]	

▽

△

Number	Heading
[SWS_EthIf_00588]	
[SWS_EthIf_00589]	
[SWS_EthIf_00590]	
[SWS_EthIf_00591]	
[SWS_EthIf_00601]	
[SWS_EthIf_00602]	
[SWS_EthIf_00604]	
[SWS_EthIf_00607]	
[SWS_EthIf_00608]	
[SWS_EthIf_00612]	
[SWS_EthIf_00614]	
[SWS_EthIf_00615]	
[SWS_EthIf_00616]	
[SWS_EthIf_00617]	
[SWS_EthIf_00618]	
[SWS_EthIf_00619]	
[SWS_EthIf_00620]	
[SWS_EthIf_00621]	
[SWS_EthIf_00626]	
[SWS_EthIf_00628]	
[SWS_EthIf_00629]	
[SWS_EthIf_00633]	
[SWS_EthIf_00634]	
[SWS_EthIf_00637]	
[SWS_EthIf_00642]	
[SWS_EthIf_00643]	
[SWS_EthIf_91137]	Definition of API function EthIf_ImmediateTransmit

Table B.9: Deleted Specification Items in R24-11

B.3 Change History of this document according to AUTOSAR Release R23-11

B.3.1 Added Specification Items in R23-11

Number	Heading
[SWS_EthIf_00102]	Definition of mandatory interfaces in module EthIf
[SWS_EthIf_00585]	
[SWS_EthIf_00586]	
[SWS_EthIf_00587]	
[SWS_EthIf_00588]	
[SWS_EthIf_00589]	
[SWS_EthIf_00590]	
[SWS_EthIf_00591]	
[SWS_EthIf_00592]	
[SWS_EthIf_00593]	
[SWS_EthIf_00600]	
[SWS_EthIf_00601]	
[SWS_EthIf_00602]	
[SWS_EthIf_00603]	
[SWS_EthIf_00604]	
[SWS_EthIf_00605]	
[SWS_EthIf_00606]	
[SWS_EthIf_00607]	
[SWS_EthIf_00608]	
[SWS_EthIf_00609]	
[SWS_EthIf_00610]	
[SWS_EthIf_00611]	
[SWS_EthIf_00612]	
[SWS_EthIf_00614]	
[SWS_EthIf_00615]	
[SWS_EthIf_00616]	
[SWS_EthIf_00617]	
[SWS_EthIf_00618]	
[SWS_EthIf_00619]	
[SWS_EthIf_00620]	
[SWS_EthIf_00621]	
[SWS_EthIf_00622]	
[SWS_EthIf_00623]	

▽

△

Number	Heading
[SWS_EthIf_00624]	
[SWS_EthIf_00625]	
[SWS_EthIf_00626]	
[SWS_EthIf_00627]	
[SWS_EthIf_00628]	
[SWS_EthIf_00629]	
[SWS_EthIf_00630]	
[SWS_EthIf_00631]	
[SWS_EthIf_00632]	
[SWS_EthIf_00633]	
[SWS_EthIf_00634]	
[SWS_EthIf_00635]	
[SWS_EthIf_00636]	
[SWS_EthIf_00637]	
[SWS_EthIf_00638]	
[SWS_EthIf_00639]	
[SWS_EthIf_00640]	
[SWS_EthIf_00641]	
[SWS_EthIf_00642]	
[SWS_EthIf_00643]	
[SWS_EthIf_00644]	
[SWS_EthIf_00645]	
[SWS_EthIf_00646]	
[SWS_EthIf_00647]	
[SWS_EthIf_00648]	
[SWS_EthIf_91023]	Definition of callback function EthIf_StreamHandleIdxStatistics
[SWS_EthIf_91024]	Definition of callback function EthIf_StreamHandleIdxConfiguration
[SWS_EthIf_91025]	Definition of API function EthIf_SetStreamHandleIdxConfiguration
[SWS_EthIf_91027]	Definition of API function EthIf_GetStreamHandleIdxStatistics
[SWS_EthIf_91062]	Definition of API function EthIf_SetPhcTime
[SWS_EthIf_91063]	Definition of API function EthIf_SetPhcCorrection
[SWS_EthIf_91064]	Definition of API function EthIf_GetPhcTime
[SWS_EthIf_91065]	Definition of API function EthIf_SetPpsSignalMode
[SWS_EthIf_91066]	Definition of API function EthIf_GetCurrentTimeTuple
[SWS_EthIf_91136]	Definiton of runtime errors in module EthIf
[SWS_EthIf_91137]	Definition of API function EthIf_ImmediateTransmit
[SWS_EthIf_91138]	Definition of API function EthIf_ReleaseRxBuffer

▽

△

Number	Heading
[SWS_EthIf_91139]	Definition of scheduled function EthIf_MainFunctionRx_<IngressQueue Processing ShortName>

Table B.10: Added Specification Items in R23-11

B.3.2 Changed Specification Items in R23-11

Number	Heading
[SWS_EthIf_00023]	Definition of imported datatypes of module EthIf
[SWS_EthIf_00085]	Definition of API function EthIf_RxIndication
[SWS_EthIf_00103]	Definition of optional interfaces in module EthIf
[SWS_EthIf_00154]	Definition of API function EthIf_GetCurrentTime
[SWS_EthIf_00155]	
[SWS_EthIf_00156]	
[SWS_EthIf_00157]	
[SWS_EthIf_00158]	
[SWS_EthIf_00245]	
[SWS_EthIf_00473]	
[SWS_EthIf_00500]	
[SWS_EthIf_00503]	Security events for EthIf
[SWS_EthIf_91026]	Definition of API function EthIf_SetRadioParams
[SWS_EthIf_91034]	Definition of API function EthIf_SetChanRxParams
[SWS_EthIf_91051]	Definition of scheduled function EthIf_MainFunctionRx_<PriorityProcessing ShortName>
[SWS_EthIf_91054]	Definition of API function EthIf_GetBufWTxParams
[SWS_EthIf_91107]	Definition of API function EthIf_GetSwitchPortMode
[SWS_EthIf_91109]	Definition of API function EthIf_SwitchPortGetLinkState
[SWS_EthIf_91111]	Definition of API function EthIf_SwitchPortGetBaudRate
[SWS_EthIf_91113]	Definition of API function EthIf_SwitchPortGetDuplexMode
[SWS_EthIf_91116]	Definition of API function EthIf_SwitchPortGetRxStats
[SWS_EthIf_91119]	Definition of API function EthIf_SwitchPortGetMacLearningMode
[SWS_EthIf_91123]	Definition of API function EthIf_ReadPortMirrorConfiguration
[SWS_EthIf_91131]	Definition of API function EthIf_RunPortCableDiagnostic
[SWS_EthIf_91132]	Definition of API function EthIf_RunCableDiagnostic

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

B.3.3 Deleted Specification Items in R23-11

none

B.3.4 Added Constraints in R23-11

Number	Heading
[SWS_EthIf_CONSTR_00002]	
[SWS_EthIf_CONSTR_00003]	
[SWS_EthIf_CONSTR_00004]	
[SWS_EthIf_CONSTR_00005]	

Table B.12: Added Constraints in R23-11

B.3.5 Changed Constraints in R23-11

none

B.3.6 Deleted Constraints in R23-11

none

B.4 Change History of this document according to AUTOSAR Release R22-11

B.4.1 Added Specification Items in R22-11

Number	Heading
[SWS_EthIf_00520]	
[SWS_EthIf_00521]	
[SWS_EthIf_00522]	
[SWS_EthIf_00523]	
[SWS_EthIf_00524]	
[SWS_EthIf_00525]	
[SWS_EthIf_00526]	
[SWS_EthIf_00531]	

▽

△

Number	Heading
[SWS_EthIf_00532]	
[SWS_EthIf_00533]	
[SWS_EthIf_00534]	
[SWS_EthIf_00535]	
[SWS_EthIf_00536]	
[SWS_EthIf_00541]	
[SWS_EthIf_00542]	
[SWS_EthIf_00543]	
[SWS_EthIf_00544]	
[SWS_EthIf_00545]	
[SWS_EthIf_00546]	
[SWS_EthIf_00547]	
[SWS_EthIf_00551]	
[SWS_EthIf_00552]	
[SWS_EthIf_00553]	
[SWS_EthIf_00554]	
[SWS_EthIf_00555]	
[SWS_EthIf_00556]	
[SWS_EthIf_00557]	
[SWS_EthIf_00560]	
[SWS_EthIf_00561]	
[SWS_EthIf_00562]	
[SWS_EthIf_00563]	
[SWS_EthIf_00564]	
[SWS_EthIf_00565]	
[SWS_EthIf_00566]	
[SWS_EthIf_00567]	
[SWS_EthIf_00568]	
[SWS_EthIf_00569]	
[SWS_EthIf_00570]	
[SWS_EthIf_00571]	
[SWS_EthIf_00572]	
[SWS_EthIf_00573]	
[SWS_EthIf_00574]	
[SWS_EthIf_00575]	
[SWS_EthIf_00576]	
[SWS_EthIf_00577]	
[SWS_EthIf_00578]	

▽

△

Number	Heading
[SWS_EthIf_00579]	
[SWS_EthIf_00580]	
[SWS_EthIf_00581]	
[SWS_EthIf_00582]	
[SWS_EthIf_00583]	
[SWS_EthIf_00584]	
[SWS_EthIf_91201]	
[SWS_EthIf_91202]	
[SWS_EthIf_91203]	
[SWS_EthIf_91204]	
[SWS_EthIf_91205]	
[SWS_EthIf_91206]	
[SWS_EthIf_91207]	
[SWS_EthIf_91208]	
[SWS_EthIf_91209]	
[SWS_EthIf_91210]	
[SWS_EthIf_91211]	
[SWS_EthIf_91212]	
[SWS_EthIf_91213]	
[SWS_EthIf_91214]	
[SWS_EthIf_91215]	
[SWS_EthIf_91216]	
[SWS_EthIf_91217]	
[SWS_EthIf_91218]	
[SWS_EthIf_91219]	
[SWS_EthIf_91220]	
[SWS_EthIf_91221]	
[SWS_EthIf_91222]	
[SWS_EthIf_91223]	
[SWS_EthIf_91224]	
[SWS_EthIf_91225]	
[SWS_EthIf_91226]	
[SWS_EthIf_91227]	
[SWS_EthIf_91228]	
[SWS_EthIf_91229]	
[SWS_EthIf_91230]	
[SWS_EthIf_91231]	
[SWS_EthIf_91232]	

▽

△

Number	Heading
[SWS_EthIf_91233]	
[SWS_EthIf_91234]	
[SWS_EthIf_91235]	
[SWS_EthIf_91236]	
[SWS_EthIf_91237]	
[SWS_EthIf_91238]	
[SWS_EthIf_CONSTR_00001]	

Table B.13: Added Specification Items in R22-11

B.4.2 Changed Specification Items in R22-11

Number	Heading
[SWS_EthIf_00017]	
[SWS_EthIf_00023]	
[SWS_EthIf_00024]	
[SWS_EthIf_00034]	
[SWS_EthIf_00035]	
[SWS_EthIf_00039]	
[SWS_EthIf_00040]	
[SWS_EthIf_00061]	
[SWS_EthIf_00067]	
[SWS_EthIf_00068]	
[SWS_EthIf_00075]	
[SWS_EthIf_00082]	
[SWS_EthIf_00085]	
[SWS_EthIf_00091]	
[SWS_EthIf_00097]	
[SWS_EthIf_00103]	
[SWS_EthIf_00104]	
[SWS_EthIf_00106]	
[SWS_EthIf_00108]	
[SWS_EthIf_00113]	
[SWS_EthIf_00115]	
[SWS_EthIf_00130]	
[SWS_EthIf_00132]	
[SWS_EthIf_00139]	

▽

△

Number	Heading
[SWS_EthIf_00147]	
[SWS_EthIf_00149]	
[SWS_EthIf_00154]	
[SWS_EthIf_00160]	
[SWS_EthIf_00166]	
[SWS_EthIf_00172]	
[SWS_EthIf_00190]	
[SWS_EthIf_00196]	
[SWS_EthIf_00214]	
[SWS_EthIf_00219]	
[SWS_EthIf_00229]	
[SWS_EthIf_00231]	
[SWS_EthIf_00232]	
[SWS_EthIf_00244]	
[SWS_EthIf_00245]	
[SWS_EthIf_00250]	
[SWS_EthIf_00263]	
[SWS_EthIf_00266]	
[SWS_EthIf_00275]	
[SWS_EthIf_00417]	
[SWS_EthIf_00421]	
[SWS_EthIf_00479]	
[SWS_EthIf_00484]	
[SWS_EthIf_00497]	
[SWS_EthIf_00498]	
[SWS_EthIf_00503]	Security events for EthIf
[SWS_EthIf_00504]	
[SWS_EthIf_91002]	
[SWS_EthIf_91003]	
[SWS_EthIf_91004]	
[SWS_EthIf_91005]	
[SWS_EthIf_91006]	
[SWS_EthIf_91007]	
[SWS_EthIf_91010]	
[SWS_EthIf_91011]	
[SWS_EthIf_91012]	
[SWS_EthIf_91013]	
[SWS_EthIf_91014]	

▽

△

Number	Heading
[SWS_EthIf_91016]	
[SWS_EthIf_91017]	
[SWS_EthIf_91018]	
[SWS_EthIf_91020]	
[SWS_EthIf_91021]	
[SWS_EthIf_91022]	
[SWS_EthIf_91026]	
[SWS_EthIf_91034]	
[SWS_EthIf_91042]	
[SWS_EthIf_91050]	
[SWS_EthIf_91051]	
[SWS_EthIf_91052]	
[SWS_EthIf_91053]	
[SWS_EthIf_91054]	
[SWS_EthIf_91055]	
[SWS_EthIf_91056]	
[SWS_EthIf_91057]	
[SWS_EthIf_91058]	
[SWS_EthIf_91059]	
[SWS_EthIf_91060]	
[SWS_EthIf_91061]	
[SWS_EthIf_91101]	
[SWS_EthIf_91102]	
[SWS_EthIf_91103]	
[SWS_EthIf_91104]	
[SWS_EthIf_91105]	
[SWS_EthIf_91106]	
[SWS_EthIf_91107]	
[SWS_EthIf_91108]	
[SWS_EthIf_91109]	
[SWS_EthIf_91110]	
[SWS_EthIf_91111]	
[SWS_EthIf_91112]	
[SWS_EthIf_91113]	
[SWS_EthIf_91114]	
[SWS_EthIf_91115]	
[SWS_EthIf_91116]	
[SWS_EthIf_91117]	

▽

△

Number	Heading
[SWS_EthIf_91118]	
[SWS_EthIf_91119]	
[SWS_EthIf_91120]	
[SWS_EthIf_91121]	
[SWS_EthIf_91122]	
[SWS_EthIf_91123]	
[SWS_EthIf_91124]	
[SWS_EthIf_91125]	
[SWS_EthIf_91126]	
[SWS_EthIf_91127]	
[SWS_EthIf_91128]	
[SWS_EthIf_91129]	
[SWS_EthIf_91130]	
[SWS_EthIf_91131]	
[SWS_EthIf_91132]	
[SWS_EthIf_91133]	
[SWS_EthIf_91134]	
[SWS_EthIf_91135]	

Table B.14: Changed Specification Items in R22-11

B.4.3 Deleted Specification Items in R22-11

none

B.4.4 Added Constraints in R22-11

none

B.4.5 Changed Constraints in R22-11

none

B.4.6 Deleted Constraints in R22-11

none