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	R23-11

Document Change History

Date	Release	Changed by	Description
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 EthernetWakeupOdDataline (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 • Editiorial 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 • Editiorial 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.

Contents

1 Introduction and functional overview 13

2 Acronyms and Abbreviations 15

3 Related documentation 16

3.1 Input documents & related standards and norms 16
3.2 Related specification 17

4 Constraints and assumptions 18

4.1 Limitations 18
4.2 Applicability to car domains 18

5 Dependencies to other modules 19

6 Requirements Tracing 20

7 Functional specification 23

7.1 Ethernet BSW stack 23
7.1.1 Indexing scheme for Ethernet controller 23
7.1.2 Indexing scheme for Ethernet switches 24
7.1.3 Ethernet Interface main function 24
7.1.4 Requirements 24
7.1.5 Configuration description 25
7.1.6 VLAN support 26
7.1.7 Wake up support 26
7.1.8 Ethernet Switch Management support 26
7.1.9 Handling of maintained Ethernet hardware 27
7.1.9.1 EthIfSwitchPortGroup 28
7.1.9.2 Switching of EthIfController and the corresponding Ethernet hardware 29
7.1.9.3 Additional Ethernet switch port handling 33
7.1.10 Communication control 34
7.1.11 Communication 35
7.1.11.1 Frame-based communication 35
7.1.11.2 PDU-based communication 35
7.1.11.3 State handling of PDUs 40
7.1.11.4 Meta data handling 41
7.1.11.5 Ingress queue handling 42
7.1.12 Global Time support 43
7.1.13 Wireless Ethernet Support 43
7.1.14 Cellular V2X Support 44
7.1.15 MACsec support 45
7.1.16 Firewall support 47
7.2 Security Events 48
7.3 Error Classification 48

7.3.1 Development Errors . . . 49
7.3.2 Runtime Errors . . . 49
7.3.3 Transient Faults . . . 49
7.3.4 Production Errors . . . 49
7.3.5 Extended Production Errors . . . 50

8 API specification 51

8.1 Imported types . . . 51
8.2 Type definitions . . . 52
8.2.1 EthIf_ConfigType . . . 52
8.2.2 EthIf_SwitchPortGroupIdxType . . . 53
8.2.3 EthIf_MeasurementIdxType . . . 53
8.2.4 EthIf_SignalQualityResultType . . . 53
8.3 Function definitions . . . 54
8.3.1 EthIf_Init . . . 54
8.3.2 EthIf_SetControllerMode . . . 55
8.3.3 EthIf_GetControllerMode . . . 55
8.3.4 EthIf_CheckWakeup . . . 56
8.3.5 EthIf_GetPhyWakeupReason . . . 57
8.3.6 EthIf_GetSwitchPortWakeupReason . . . 58
8.3.7 EthIf_GetPhysAddr . . . 59
8.3.8 EthIf_SetPhysAddr . . . 60
8.3.9 EthIf_UpdatePhysAddrFilter . . . 61
8.3.10 EthIf_GetPortMacAddr . . . 62
8.3.11 EthIf_GetArlTable . . . 63
8.3.12 EthIf_GetCtrlIdxList . . . 64
8.3.13 EthIf_GetVlanId . . . 64
8.3.14 EthIf_GetAndResetMeasurementData . . . 65
8.3.15 EthIf_StoreConfiguration . . . 66
8.3.16 EthIf_ResetConfiguration . . . 67
8.3.17 EthIf_GetCurrentTime . . . 68
8.3.18 EthIf_GetCurrentTimeTuple . . . 69
8.3.19 EthIf_SetPhcTime . . . 70
8.3.20 EthIf_SetPhcCorrection . . . 71
8.3.21 EthIf_GetPhcTime . . . 72
8.3.22 EthIf_SetPpsSignalMode . . . 73
8.3.23 EthIf_EnableEgressTimeStamp . . . 75
8.3.24 EthIf_GetEgressTimeStamp . . . 75
8.3.25 EthIf_GetIngressTimeStamp . . . 76
8.3.26 EthIf_SwitchPortGroupRequestMode . . . 77
8.3.27 EthIf_StartAllPorts . . . 79
8.3.28 EthIf_SetSwitchMgmtInfo . . . 79
8.3.29 EthIf_GetRxMgmtObject . . . 80
8.3.30 EthIf_GetTxMgmtObject . . . 81
8.3.31 EthIf_SwitchEnableTimeStamping . . . 81
8.3.32 EthIf_VerifyConfig . . . 82

8.3.33 EthIf_SetForwardingMode . . . 83
8.3.34 EthIf_GetTrcvSignalQuality . . . 84
8.3.35 EthIf_GetSwitchPortSignalQuality . . . 85
8.3.36 EthIf_ClearTrcvSignalQuality . . . 86
8.3.37 EthIf_ClearSwitchPortSignalQuality . . . 86
8.3.38 EthIf_SetPhyTestMode . . . 87
8.3.39 EthIf_SetPhyLoopbackMode . . . 88
8.3.40 EthIf_SetPhyTxMode . . . 89
8.3.41 EthIf_GetCableDiagnosticsResult . . . 90
8.3.42 EthIf_GetPhyIdentifier . . . 91
8.3.43 EthIf_GetBufWRxParams . . . 92
8.3.44 EthIf_GetBufWTxParams . . . 93
8.3.45 EthIf_SetBufWTxParams . . . 94
8.3.46 EthIf_SetRadioParams . . . 95
8.3.47 EthIf_SetChanRxParams . . . 96
8.3.48 EthIf_SetChanTxParams . . . 97
8.3.49 EthIf_GetChanRxParams . . . 98
8.3.50 EthIf_ProvideTxBuffer . . . 99
8.3.51 EthIf_Transmit . . . 101
8.3.52 EthIf_ImmediateTransmit . . . 102
8.3.53 EthIf_ReleaseRxBuffer . . . 102
8.3.54 EthIf_GetVersionInfo . . . 103
8.3.55 EthIf_GetSwitchPortMode . . . 104
8.3.56 EthIf_GetTransceiverMode . . . 104
8.3.57 EthIf_SwitchPortGetLinkState . . . 105
8.3.58 EthIf_TransceiverGetLinkState . . . 106
8.3.59 EthIf_SwitchPortGetBaudRate . . . 106
8.3.60 EthIf_TransceiverGetBaudRate . . . 107
8.3.61 EthIf_SwitchPortGetDuplexMode . . . 108
8.3.62 EthIf_TransceiverGetDuplexMode . . . 108
8.3.63 EthIf_SwitchPortGetCounterValues . . . 109
8.3.64 EthIf_SwitchPortGetRxStats . . . 110
8.3.65 EthIf_SwitchPortGetTxStats . . . 110
8.3.66 EthIf_SwitchPortGetTxErrorCounterValues . . . 111
8.3.67 EthIf_SwitchPortGetMacLearningMode . . . 111
8.3.68 EthIf_GetSwitchPortIdentifier . . . 112
8.3.69 EthIf_GetSwitchIdentifier . . . 113
8.3.70 EthIf_WritePortMirrorConfiguration . . . 113
8.3.71 EthIf_ReadPortMirrorConfiguration . . . 114
8.3.72 EthIf_DeletePortMirrorConfiguration . . . 115
8.3.73 EthIf_GetPortMirrorState . . . 116
8.3.74 EthIf_SetPortMirrorState . . . 116
8.3.75 EthIf_SetPortTestMode . . . 117
8.3.76 EthIf_SetPortLoopbackMode . . . 118
8.3.77 EthIf_SetPortTxMode . . . 118
8.3.78 EthIf_GetPortCableDiagnosticsResult . . . 119

8.3.79 EthIf_RunPortCableDiagnostic . . . 120
8.3.80 EthIf_RunCableDiagnostic . . . 120
8.3.81 EthIf_SwitchGetCfgDataRaw . . . 121
8.3.82 EthIf_SwitchGetCfgDataInfo . . . 122
8.3.83 EthIf_SwitchPortGetMaxFIFOBufferFillLevel . . . 122
8.3.84 EthIf_TransceiverGetMacMethod . . . 123
8.3.85 EthIf_EthGetSpiStatus . . . 124
8.3.86 EthIf_GetBufCV2xPC5RxParams . . . 125
8.3.87 EthIf_GetBufCV2xPC5TxParams . . . 126
8.3.88 EthIf_SetBufCV2xPC5TxParams . . . 127
8.3.89 EthIf_GetChanCV2xPC5TxParams . . . 128
8.3.90 EthIf_SwitchMacSecUpdateSecY . . . 129
8.3.91 EthIf_MacSecUpdateSecY . . . 130
8.3.92 EthIf_SwitchMacSecUpdateSecYNotification . . . 130
8.3.93 EthIf_MacSecUpdateSecYNotification . . . 131
8.3.94 EthIf_SwitchMacSecInitRxSc . . . 131
8.3.95 EthIf_MacSecInitRxSc . . . 132
8.3.96 EthIf_SwitchMacSecResetRxSc . . . 133
8.3.97 EthIf_MacSecResetRxSc . . . 133
8.3.98 EthIf_SwitchMacSecAddTxSa . . . 134
8.3.99 EthIf_MacSecAddTxSa . . . 135
8.3.100 EthIf_SwitchMacSecAddTxSaNotification . . . 135
8.3.101 EthIf_MacSecAddTxSaNotification . . . 136
8.3.102 EthIf_SwitchMacSecUpdateTxSa . . . 137
8.3.103 EthIf_MacSecUpdateTxSa . . . 137
8.3.104 EthIf_SwitchMacSecDeleteTxSa . . . 138
8.3.105 EthIf_MacSecDeleteTxSa . . . 139
8.3.106 EthIf_SwitchMacSecAddRxSa . . . 139
8.3.107 EthIf_MacSecAddRxSa . . . 140
8.3.108 EthIf_SwitchMacSecAddRxSaNotification . . . 141
8.3.109 EthIf_MacSecAddRxSaNotification . . . 141
8.3.110 EthIf_SwitchMacSecUpdateRxSa . . . 142
8.3.111 EthIf_MacSecUpdateRxSa . . . 143
8.3.112 EthIf_SwitchMacSecDeleteRxSa . . . 143
8.3.113 EthIf_MacSecDeleteRxSa . . . 144
8.3.114 EthIf_SwitchMacSecGetTxSaNextPn . . . 145
8.3.115 EthIf_MacSecGetTxSaNextPn . . . 145
8.3.116 EthIf_SwitchMacSecGetMacSecStats . . . 146
8.3.117 EthIf_MacSecGetMacSecStats . . . 147
8.3.118 EthIf_SwitchMacSecGetMacSecStatsNotification . . . 147
8.3.119 EthIf_MacSecGetMacSecStatsNotification . . . 148
8.3.120 EthIf_SwitchMacSecOperational . . . 148
8.3.121 EthIf_MacSecOperational . . . 149
8.3.122 EthIf_SwitchMacSecSetControlledPortEnabled . . . 149
8.3.123 EthIf_MacSecSetControlledPortEnabled . . . 150
8.3.124 EthIf_GetStreamHandleIdxStatistics . . . 151

8.3.125 EthIf_SetStreamHandleIdxConfiguration . . . 151
8.4 Callback notifications . . . 152
8.4.1 EthIf_RxIndication . . . 152
8.4.2 EthIf_TxConfirmation . . . 153
8.4.3 EthIf_CtrlModeIndication . . . 154
8.4.4 EthIf_TrcvModeIndication . . . 154
8.4.5 EthIf_SwitchPortModeIndication . . . 155
8.4.6 EthIf_SleepIndication . . . 155
8.4.7 EthIf_StreamHandleIdxConfiguration . . . 156
8.4.8 EthIf_StreamHandleIdxStatistics . . . 156
8.5 Scheduled functions . . . 157
8.5.1 EthIf_MainFunctionRx . . . 157
8.5.2 EthIf_MainFunctionRx_<PriorityProcessing ShortName> . . . 158
8.5.3 EthIf_MainFunctionRx_<IngressQueueProcessing ShortName> . . . 158
8.5.4 EthIf_MainFunctionTx . . . 159
8.5.5 EthIf_MainFunctionState . . . 159
8.6 Expected interfaces . . . 161
8.6.1 Mandatory interfaces . . . 161
8.6.2 Optional interfaces . . . 161
8.6.3 Configurable interfaces . . . 165

9 Sequence diagrams 168

9.1 Initialization . . . 168
9.2 Communication Initialization . . . 169
9.3 Switch Initialization . . . 170
9.4 Data Transmission . . . 171
9.5 Data Reception . . . 173
9.6 Link State Change . . . 174
9.7 Link State Change without Port Groups . . . 175
9.8 Link State Change with Port Groups . . . 176
9.9 Link State Change with Port Groups and Partial Network Cluster . . . 177
9.10 Switch Management support . . . 178

10 Configuration specification 180

10.1 How to read this chapter . . . 180
10.2 Containers and configuration parameters . . . 180
10.2.1 EthIf . . . 188
10.2.2 EthIfGeneral . . . 188
10.2.3 EthIfConfigSet . . . 199
10.2.4 EthIfController . . . 199
10.2.5 EthIfFrameOwnerConfig . . . 203
10.2.5.1 EthIfFrameOwnerPdu . . . 205
10.2.6 EthIfPhysController . . . 209
10.2.7 EthIfPhysCtrlRxMainFunctionIngressQueueProcessing . . . 212
10.2.8 EthIfClkUnit . . . 214
10.2.9 EthIfRxIndicationConfig . . . 215

10.2.10 EthIfSwitch . . . 216
10.2.11 EthIfSwitchPortGroup . . . 216
10.2.12 EthIfTransceiver . . . 218
10.2.13 EthIfTrcvLinkStateChgConfig . . . 220
10.2.14 EthIfTxConfirmationConfig . . . 220
10.2.15 EthIfSecurityEventRefs . . . 221
10.3 Published Information . . . 223

A Not applicable requirements 224

B Change History 225

B.1 Change History of this document according to AUTOSAR Release R22-11 . . . 225
B.1.1 Added Specification Items in R22-11 . . . 225
B.1.2 Changed Specification Items in R22-11 . . . 227
B.1.3 Deleted Specification Items in R22-11 . . . 231
B.1.4 Added Constraints in R22-11 . . . 231
B.1.5 Changed Constraints in R22-11 . . . 231
B.1.6 Deleted Constraints in R22-11 . . . 231
B.2 Change History of this document according to AUTOSAR Release R23-11 . . . 231
B.2.1 Added Specification Items in R23-11 . . . 231
B.2.2 Changed Specification Items in R23-11 . . . 233
B.2.3 Deleted Specification Items in R23-11 . . . 234
B.2.4 Added Constraints in R23-11 . . . 234
B.2.5 Changed Constraints in R23-11 . . . 235
B.2.6 Deleted Constraints in R23-11 . . . 235

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).⌋*()*

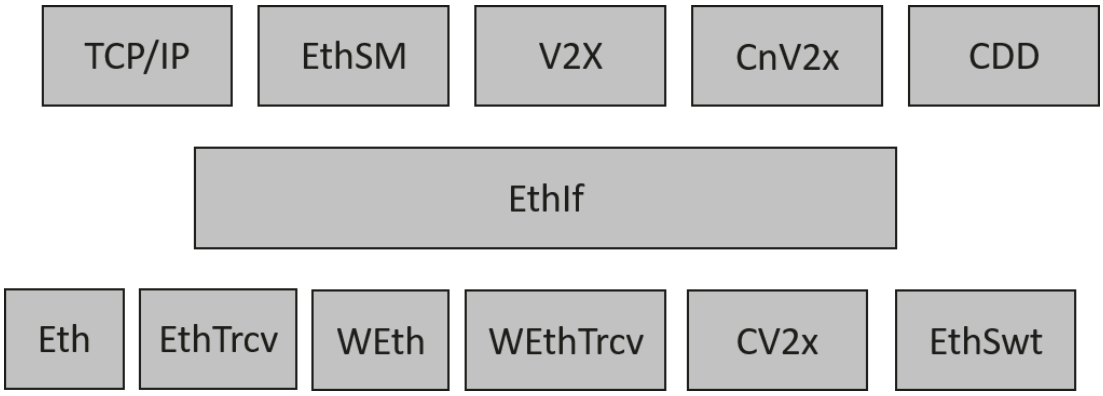

Figure 1.1: Ethernet stack module overview

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]

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_SRS_BSWGeneral

[13] Requirements on Ethernet Support in AUTOSAR
AUTOSAR_CP_SRS_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-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 and one or more Ethernet Transceiver Driver.

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_00581] [SWS_EthIf_00582]
[FO_RS_MACsec_-00004]	Configure which Ethernet ports use MACsec	[SWS_EthIf_00561] [SWS_EthIf_00562]
[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]
[RS_Ids_00810]	Basic SW security events	[SWS_EthIf_00502] [SWS_EthIf_00503]
[SRS_BSW_00101]	The Basic Software Module shall be able to initialize variables and hardware in a separate initialization function	[SWS_EthIf_00304] [SWS_EthIf_00306]

▽

△

Requirement	Description	Satisfied by
[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_00334]	All Basic Software Modules shall provide an XML file that contains the meta data	[SWS_EthIf_CONSTR_00002] [SWS_EthIf_CONSTR_00003]
[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_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]
[SRS_BSW_00369]	All AUTOSAR Basic Software Modules shall not return specific development error codes via the API	[SWS_EthIf_00304] [SWS_EthIf_00306]
[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_00601] [SWS_EthIf_00602] [SWS_EthIf_00603] [SWS_EthIf_00604] [SWS_EthIf_00607] [SWS_EthIf_00608] [SWS_EthIf_00609] [SWS_EthIf_00612] [SWS_EthIf_00620] [SWS_EthIf_00621] [SWS_EthIf_00622] [SWS_EthIf_00626] [SWS_EthIf_00627] [SWS_EthIf_00628] [SWS_EthIf_00629] [SWS_EthIf_00633] [SWS_EthIf_00634] [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]
[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_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_00125]	The Ethernet Switch Driver shall support switch frame management	[SWS_EthIf_91003] [SWS_EthIf_91007]
[SRS_Eth_00156]	The Ethernet Interface shall provide indication for a received sleep request.	[SWS_EthIf_00497] [SWS_EthIf_00499] [SWS_EthIf_91006]

▽

△

Requirement	Description	Satisfied by
[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]
[SRS_Eth_00169]	Ethernet Interface upper layer PDU based communication	[SWS_EthIf_00085] [SWS_EthIf_91137] [SWS_EthIf_91138] [SWS_EthIf_CONSTR_00002] [SWS_EthIf_CONSTR_00003]
[SRS_Eth_00170]	Ethernet Interface scheduling a subset of ingress queues	[SWS_EthIf_00648] [SWS_EthIf_91139] [SWS_EthIf_CONSTR_00004] [SWS_EthIf_CONSTR_00005]
[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]

Table 6.1: RequirementsTracing

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.

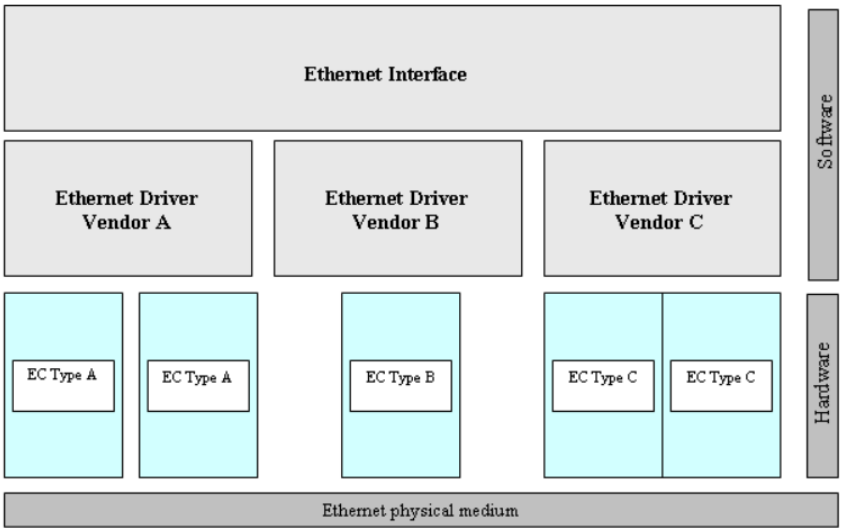

Figure 7.1: Basic Structure of the Enthernet BSW Stack

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.

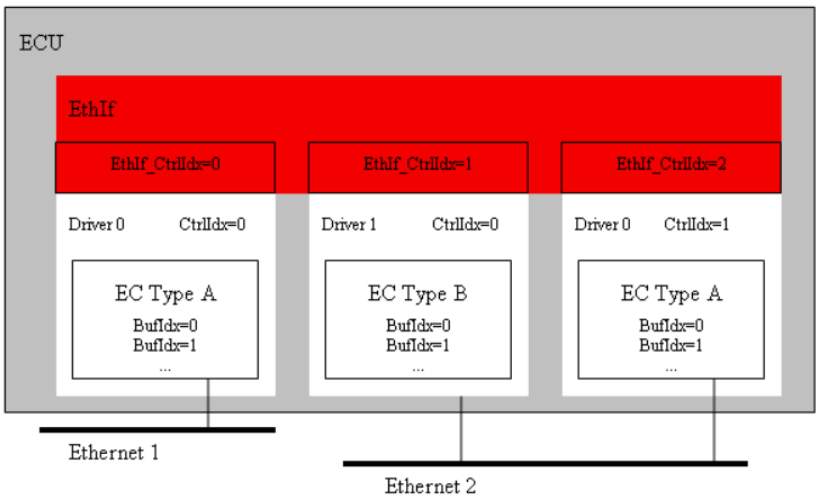

Figure 7.2: Ethernet Interface controller indexing scheme

[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 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.⌋*()*

7.1.4 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.⌋*()*

[SWS_EthIf_00008] ⌈In case development error detection is enabled for the Ethernet Interface module: The Ethernet Interface module shall check API parameters for validity and report detected errors to the DET.⌋*()*

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.⌋*()*

7.1.5 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.

7.1.6 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.⌋*()*

7.1.7 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.

7.1.8 Ethernet Switch Management support

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.

7.1.9 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_SwitchPortGroupRequestMode`. 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_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 ACITVE-request or switch off the corresponding Ethernet Hardware for a DOWN-request.

7.1.9.1 EthIfSwitchPortGroup

The Ethernet Interface supports the grouping of Ethernet switch ports (EthIfSwitch-PortGroup). The request (either ACITVE or DOWN) will be handled and rated by the Ethernet Interface. The Ethernet Interface has to decide either to put the EthifSwitch-PotGroup 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 7.1.9.2 and 8.3.26.

7.1.9.1.1 Link state accumulation of EthIfSwitchPortGroup

The Ethernet Interface need to know the actual link state of the EthIfSwitchPortGroups. The link state for an EthIfSwitchPortGroup is computed over all link states of the Eth-SwtPorts 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 is the actual state of the EthIfSwitchPortGroup. The actual state of the EthIfSwitchPortGroup. The actual state of EthIfSwitchPortGroups referenced by an EthIfController is reported to the EthSM by calling `EthSM_TrcvLinkStateChg`. The actual state of EthIfSwitchPort-Groups which are not referenced by any EthIfController is reported to the BswM by calling `BswM_EthIf_PortGroupLinkStateChg`.

[SWS_EthIf_00259] ⌈The link state for an EthIfSwitchPortGroup is computed over all link states of the EthSwtPorts which are referenced by the EthIfSwitchPortGroup. Its status is `ETHTRCV_LINK_STATE_DOWN` (link down) if one of the following conditions is met:

- Referenced EthSwtPort with the role "host port" or the role "up link port" has link down state
- All referenced EthSwtPort without a role have link down state

Otherwise its accumulated link state is `ETHTRCV_LINK_STATE_ACTIVE` (link up).⌋*()*

[SWS_EthIf_00260] ⌈If the EthIfCtrl references a EthIfSwitch but no port group is configured, the EthIf shall indicate the link state of the host port to the EthSM by calling `EthSM_TrcvLinkStateChg` for the EthIfController when the link state changes.⌋*()*

[SWS_EthIf_00261] ⌈In case a EthIfSwitchPortGroup is not connected to any EthIfController, the 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).⌋*()*

[SWS_EthIf_00262] ⌈In case a EthIfSwitchPortGroup is connected to a EthIfController, the EthIf shall indicate the accumulated link state of the EthIfSwitchPortGroup to the EthSM by calling `EthSM_TrcvLinkStateChg` for the EthIfController when the link state changes (refer to [SWS_EthIf_00259] for link state accumulation).⌋*()*

7.1.9.2 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_ACITVE` 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_ACITVE` 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] ⌈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

⌋*(SRS_Eth_00157)*

[SWS_EthIf_00478] ⌈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

⌋*(SRS_Eth_00157)*

[SWS_EthIf_00264] ⌈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

⌋*(SRS_Eth_00157)*

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] ⌈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

⌋*(SRS_Eth_00157)*

[SWS_EthIf_00480] ⌈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

⌋*(SRS_Eth_00157)*

[SWS_EthIf_00481] ⌈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

⌋(*SRS_Eth_00157*)

[SWS_EthIf_00482] ⌈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`

⌋(*SRS_Eth_00157*)

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] ⌈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.⌋(*SRS_Eth_00157*)

[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 EthIfSwitchPortGroup 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.

7.1.9.3 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 7.1.9.2.

7.1.10 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] ⌈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_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`.⌋*(SRS_Eth_00157)*

Note: The transmission related APIs (see [SWS_EthIf_00075] and [SWS_EthIf_00067]) will only forward transmission requests, if the stored communication mode is `ETH_MODE_ACTIVE` or `ETH_MODE_ACTIVE_WITH_WAKEUP_REQUEST`.

7.1.11 Communication

The Ethernet Interface support two approaches to transfer data from the lower layer the upper layer and vice versa. The first is a frame-based and the second is a PDU-based communication approach.

7.1.11.1 Frame-based communication

The frame-based communication approach interchanges frame-specific information with specific API parameters. For transmission a two-way API call is used :

1. Upper layer call `EthIf_ProvideTxBuffer` to request an egress buffer at the Ethernet driver according the given priority. After return, the upper layer copy data to the provided buffer
2. Upper layer call `EthIf_Transmit` to request the Ethernet driver to transmit the content of the egress buffer

Specification for transmission can be found in subsection 8.3.51 and subsection 8.3.50

For reception the `EthIf_RxIndication` is used and forwarded to the configured upper layer via the upper layer provided RxIndication API (e.g. `TcpIp_RxIndication`). Specification for reception can be found in subsection 8.4.1.

In both cases the EthIf module act as path-through module. This approach is used to serve [2, SWS TcpIp module] and [15, SWS TimeSyncOverEthernet module].

7.1.11.2 PDU-based communication

The PDU-based 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 subsubsection 7.1.11.4)

Note: The EthIf module interchange PDUs with the upper layers (e.g. [17, SWS IEEE1722Tp module]) via the [18, SWS L-SDU router module].

For frame-based communication it is sufficient to configure `EthIfFrameOwnerConfig`s, where each owner is assigned to a `EthIfFrameType`. `EthIfFrameType` denote a specific type of frame (e.g. 0x22F01 (AVTP EtherType)). The neighboring parameter `EthIfOwner` defines the RxIndication functions. The EthIf module provide the possibility to configure additionally `EthIfFrameOwnerPdu`s to denote PDU-based communication. One `EthIfFrameOwnerPdu` could aggregate multiple `EthIfFrameOwnerPduPoolEntry`s. Multiple `EthIfFrameOwnerPduPoolEntry`s `EthIfFrameOwnerPdu` form a so-called "PDU-pool". A `EthIfFrameOwnerPdu`

could either have EthIfFrameOwnerPduDirection set to RX or TX. RX define the communication direction from the EthIf module to the upper layer. TX define the communication direction from the upper layer to the EthIf module.

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 one or multiple EthIfFrameType of the config. If the matches have EthIfFrameOwnerPdus configured, then EthIf has to search for an available EthIfFrameOwnerPduPoolEntry, and if one EthIfFrameOwnerPduPoolEntry is available, then this EthIfFrameOwnerPduPoolEntry 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. The destination upper layer perform an reception processing. Optionally, if the reception processing is finalized, the upper layer could explicitly indicate to release EthIfFrameOwnerPduPoolEntry 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 EthCtrlConfigInressQueue element is locked, and consequently also the used EthIfFrameOwnerPduPoolEntry. For further receptions another EthIfFrameOwnerPduPoolEntry of the corresponding PDU-pool 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 EthCtrlConfigInressQueue element need to be locked until the upper layer indicate to release EthCtrlConfigInressQueue element. If a global PDU is configured with KeepLocalPduBuffer set to FALSE, the EthIf module call directly Eth_ReleaseRxBuffer after the RxIndication function call returns.

Please note: since the Eth driver is not PDU-aware, the Eth driver provide frame specific API parameter via EthIf_RxIndication.

[SWS_EthIf_00637]{DRAFT} ⌈If EthIf_RxIndication is called and the following condition are true:

- the given FrameType match to EthIfFrameType of a configured EthIfFrameOwnerConfig
- the matching EthIfFrameOwnerConfig has an EthIfFrameOwnerPdu configured
- this EthIfFrameOwnerPdu has at least one EthIfFrameOwnerPduPoolEntry configured with EthIfFrameOwnerPduDirection set to RX
- this EthIfFrameOwnerPduPoolEntry is in state PDU_AVAILABLE

- the VLAN-ID of the given Ethernet frame match to the referenced EthIfController

then the EthIf shall perform the following action. Otherwise EthIf shall abort processing of the reception indication:

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

⌋(*SRS_BSW_00350*, *SRS_BSW_00386*)

[SWS_EthIf_00638]{DRAFT} ⌈If a reception indication processing is aborted, then EthIf shall call Eth_ReleaseRxBuffer with the given RxHandleId.⌋(*SRS_BSW_00350*, *SRS_BSW_00386*)

[SWS_EthIf_00639]{DRAFT} ⌈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 EthIfFrameOwnerPduPoolEntry 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 EthIfFrameOwnerPduPoolEntry
- call Eth_ReleaseRxBuffer with RxHandleId mapped to the given RxPduId

⌋(*SRS_BSW_00350*, *SRS_BSW_00386*)

[SWS_EthIf_00640]{DRAFT} ⌈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.⌋(*SRS_BSW_00350*, *SRS_BSW_00386*)

[SWS_EthIf_00641]{DRAFT} ⌈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 EthIfFrameOwnerPduPoolEntry is in state PDU_IN_USE

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

- set the affected EthIfFrameOwnerPduPoolEntry 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 EthIfFrameOwnerPduPoolEntry
- call Eth_ReleaseRxBuffer with RxHandleId mapped to the given RxPduId

⌋(*SRS_BSW_00350*, *SRS_BSW_00386*)

Transmission

If EthIf is requested via a call of EthIf_ImmediateTransmit to transmit data, the EthIf need to check if given TxPduId is available and ,if so, transform the given PduInfoPtr together with the given meta data to a frame-based API call of Eth_ImmediateTransmit. A call of EthIf_TxConfirmation would release the EthIfFrameOwnerPduPoolEntry with TxPduId and forward the confirmation to the upper layer.

[SWS_EthIf_00642]{DRAFT} ⌈If EthIf_ImmediateTransmit is called and the following conditions are true:

- the given TxPduId match to a EthIfFrameOwnerPduId aggregated by a EthIfFrameOwnerPduPoolEntry of a configured EthIfFrameOwnerPdu
- this EthIfFrameOwnerPduPoolEntry belongs to an EthIfFrameOwnerPdu with EthIfFrameOwnerPduDirection set to TX
- the referenced PDU of the matching EthIfFrameOwnerPduId is in state PDU_AVAILABLE

then the EthIf shall perform the following actions as preparation for a call of Eth_ImmediateTransmit. Otherwise the function shall return with E_NOT_OK:

- set the CtrlIdx parameter to the corresponding physical EthCtrl which belongs to the referencing EthIfController

- set the TxHandleId parameter to the matching EthIfFrameOwnerPduId of the used EthIfFrameOwnerPduPoolEntry
- set priority to configured value of EthIfFrameOwnerTxPriority of the EthIfFrameOwnerPdu which aggregates the used EthIfFrameOwnerPduPoolEntry
- create a list-element-struct of type ListElemStructType according to [SWS_EthIf_00643] 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 SduDataLength given with the received PduInfoPtr
- call Eth_ImmediateTransmit with CtrlIdx, TxHandleId, priority, HeaderListPtr, PayloadPtr and PayloadLength set to values as described by the previous steps

⌋(*SRS_BSW_00350, SRS_BSW_00386*)

[SWS_EthIf_00643]{DRAFT} ⌈If EthIf has to create a list-element-struct of type ListElemStructType due to a immediate transmission request and the given TxPduId match to EthIfFrameOwnerPduId 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 according EthIfFrameOwnerPdu
- use the EthIfFrameType of the EthIfFrameOwnerConfig which aggregates the EthIfFrameOwnerPdu that refers to the used EthIfFrameOwnerPduPoolEntry
- derive the VLAN-ID of the EthIfController which is referenced by the EthIfFrameOwnerPdu of the used EthIfFrameOwnerPduPoolEntry
- use the configured priority value of EthIfFrameOwnerTxPriority of EthIfFrameOwnerPdu which aggregates the used EthIfFrameOwnerPduPoolEntry
- 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 HeaderListPtr from MetaDataItem LISTELEM_PTR and

set NextListElemPtr of the created list-element-struct to the address of the consumed HeaderListPtr. Otherwise set NextListElemPtr of the created list-element-struct to NUL_PTR.

⌋(*SRS_BSW_00350*, *SRS_BSW_00386*)

[SWS_EthIf_00600]{DRAFT} ⌈If Eth_ImmediateTransmit 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 Eth_ImmediateTransmit, the buffer for local produced data of the affected PDU shall be released.⌋(*SRS_BSW_00350*, *SRS_BSW_00386*)

[SWS_EthIf_00644]{DRAFT} ⌈If EthIf_TxConfirmation is called and the following conditions are true:

- the given BufIdx and CtrlIdx refer to a EthIfFrameOwnerPduPoolEntry (given in previous call of Eth_ImmediateTransmit as TxHandleId)
- the affected EthIfFrameOwnerPduPoolEntry 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 EthIfFrameOwnerPduPoolEntry to state PDU_AVAILABLE
- call LSduR_EthIfTxConfirmation with TxPduId set to PDU-ID referenced EthIfFrameOwnerPduPoolEntry

⌋(*SRS_BSW_00350*, *SRS_BSW_00386*)

7.1.11.3 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]{DRAFT} ⌈The EthIf module shall maintain for each PDU of all configured EthIfFrameOwnerPduPoolEntrys two states: state PDU_AVAILABLE and state PDU_IN_USE⌋(*SRS_BSW_00350*, *SRS_BSW_00386*)

[SWS_EthIf_00646]{DRAFT} ⌈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.

⌋(*SRS_BSW_00350*, *SRS_BSW_00386*)

[SWS_EthIf_00647]{DRAFT} ⌈If the EthIf module is requested to transmit data and the function call Eth_ImmediateTransmit returns with E_NOT_OK, then the IEEE1722Tp module shall set the state of the affected PDU to PDU_AVAILABLE.⌋ (*SRS_BSW_00350*, *SRS_BSW_00386*)

7.1.11.4 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

7.1.11.4.1 Meta data types

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

[SWS_EthIf_CONSTR_00002]{DRAFT} ⌈A PDU which refer to an EthIfFrameOwnerPdu where the EthIfFrameOwnerPduDirection is set to RX, shall have no other MetaDataItem of MetaDataItemType configured than:

- TIMETUPLE_TYPE_PTR

⌋(*SRS_Eth_00169, SRS_BSW_00334*)

[SWS_EthIf_CONSTR_00003]{DRAFT} ⌈A PDU which refer to an `EthIfFrameOwnerPdu` where the `EthIfFrameOwnerPduDirection` is set to `TX`, shall have no other `MetaDataItem` of `MetaDataItemType` configured than:

- `ETHERNET_MAC_64`
- `TIMETUPLE_TYPE_PTR`
- `LISTELEM_PTR`

⌋(*SRS_Eth_00169, SRS_BSW_00334*)

7.1.11.5 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 `EthIfPhysCtrlRxMainFunctionIngressProcessing`, 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 `EthIfPhysCtrlRxMainFunctionIngressProcessing`. A `EthIfPhysCtrlRxMainFunctionIngressProcessing` could reference one ingress queue via `EthIfCanXLCtrlRxIngressFifoRef` or `EthIfPhysCtrlRxIngressQueueRef`. Along with this reference a specific main function is gener-

ated (see EthIf_MainFunctionRx_<IngressQueueProcessing ShortName>), where the ingress queue handler is implemented.

[SWS_EthIf_CONSTR_00004]{DRAFT} ⌈If a EthIfPhysController have EthIfPhysCtrlRxMainFunctionIngressProcessing 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⌋ *(SRS_Eth_00170)*

[SWS_EthIf_CONSTR_00005]{DRAFT} ⌈If a EthIfPhysController have EthIfPhysCtrlRxMainFunctionIngressProcessing configured, then the referenced ingress queues via EthIfPhysCtrlRxIngressQueueRef or EthIfCanXLCtrlRxIngressFifoRef shall be handled by the corresponding EthIf_MainFunctionRx_<IngressQueueProcessing ShortName>⌋*(SRS_Eth_00170)*

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]{DRAFT} ⌈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

⌋*(SRS_Eth_00170)*

7.1.12 Global Time support

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

7.1.13 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].

7.1.14 Cellular V2X Support

[SWS_EthIf_00520]{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].

7.1.15 MACsec support

[SWS_EthIf_00560]{DRAFT} ⌈The Ethernet Interface shall support MACsec as a SW implementation as specified in [23].⌋*(FO_RS_MACsec_00001)*

[SWS_EthIf_00561]{DRAFT} ⌈The Ethernet Interface shall support configuring which Ethernet Interface Controllers are MACsec protected.⌋*(FO_RS_MACsec_00004)*

[SWS_EthIf_00562]{DRAFT} ⌈The Ethernet Interface shall support configuring per Ethernet Interface Controller the MACsec Entity to use (per SW or HW i.e., offloaded).⌋*(FO_RS_MACsec_00004)*

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

[SWS_EthIf_00563]{DRAFT} ⌈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.⌋*(FO_RS_MACsec_00007)*

[SWS_EthIf_00564]{DRAFT} ⌈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 [23].

⌋*(FO_RS_MACsec_00009)*

[SWS_EthIf_00565]{DRAFT} ⌈The MACsec entity per SW of the Ethernet Interface shall support "Integrity only" as well as "Integrity with Confidentiality" for all supported ciphers.⌋*(FO_RS_MACsec_00010)*

[SWS_EthIf_00566]{DRAFT} ⌈The MACsec entity per SW of the Ethernet Interface shall support MAC Security TAG (SecTAG) as defined in [23]. The SecTAG shall convey:

- TAG Control Information (TCI)
- Association Number (AN)
- Short Length (SL)
- Packet Number (PN)
- Secure Channel Identifier (SCI) - Optional

⌋*(FO_RS_MACsec_00011)*

[SWS_EthIf_00567]{DRAFT} ⌈The MACsec entity per SW of the Ethernet Interface shall support MACsec EtherType as defined in [23].⌋*(FO_RS_MACsec_00012)*

[SWS_EthIf_00568]{DRAFT} ⌈The MACsec entity per SW of the Ethernet Interface shall support TAG Control Information (TCI) as defined in [23]. The TCI shall be encoded in the SecTAG.⌋*(FO_RS_MACsec_00011)*

[SWS_EthIf_00569]{DRAFT} ⌈The MACsec entity per SW of the Ethernet Interface shall support Association Number (AN) as defined in [23]. The AN shall be encoded in the SecTAG.⌋*(FO_RS_MACsec_00011)*

[SWS_EthIf_00570]{DRAFT} ⌈The MACsec entity per SW of the Ethernet Interface shall support Short Length (SL) as defined in [23]. The SL shall be encoded in the SecTAG.⌋*(FO_RS_MACsec_00011)*

[SWS_EthIf_00571]{DRAFT} ⌈The MACsec entity per SW of the Ethernet Interface shall support Packet Number (PN) with 32 least significant bits, as defined in [23]. The PN shall be encoded in the SecTAG.⌋*(FO_RS_MACsec_00011)*

[SWS_EthIf_00572]{DRAFT} ⌈The MACsec entity per SW of the Ethernet Interface shall support Extended Packet Number (XPN) as defined in [23]. The XPN extends the PN to 64 bits.⌋*(FO_RS_MACsec_00017)*

[SWS_EthIf_00573]{DRAFT} ⌈The MACsec entity per SW of the Ethernet Interface shall support Secure Channel Identifier (SCI), as defined in [23]. The SCI may be encoded in the SecTAG if SCI is required to be sent.⌋*(FO_RS_MACsec_00018)*

[SWS_EthIf_00574]{DRAFT} ⌈The MACsec entity per SW of the Ethernet Interface shall support Secure Data as defined in [23].⌋*(FO_RS_MACsec_00019)*

[SWS_EthIf_00575]{DRAFT} ⌈The MACsec entity per SW of the Ethernet Interface shall support Integrity check value (ICV) as defined in [23]. 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.⌋*(FO_RS_MACsec_00020)*

[SWS_EthIf_00576]{DRAFT} ⌈The MACsec entity per SW of the Ethernet Interface shall support a protect function as specified in [23].⌋*(FO_RS_MACsec_00021)*

[SWS_EthIf_00577]{DRAFT} ⌈The MACsec entity per SW of the Ethernet Interface shall support a validation function as specified in [23].⌋*(FO_RS_MACsec_00022)*

[SWS_EthIf_00578]{DRAFT} ⌈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

⌋*(FO_RS_MACsec_00032)*

[SWS_EthIf_00579]{DRAFT} ⌈The MACsec entity per SW of the Ethernet Interface shall support a validation function for MACsec ICV.⌋*(FO_RS_MACsec_00033)*

[SWS_EthIf_00580]{DRAFT} ⌈The MACsec entity per SW of the Ethernet Interface shall support a generation function for MACsec ICV.⌋*(FO_RS_MACsec_00034)*

[SWS_EthIf_00581]{DRAFT} ⌈The Ethernet Interface Module shall share the MACsec Operational status between Ethernet Interface Controllers sharing a physical or virtual controlled port. An Ethernet Interface controller shall trigger the MKA Module to start the MKA sequence in a port with MKA_LinkStateChange after receiving the "Mode Indication" from the Switch or Transceiver with the corresponding function `EthIf_SwitchPortModeIndication` or `EthIf_TrcvModeIndication`.⌋*(FO_RS_MACsec_00002)*

[SWS_EthIf_00582]{DRAFT} ⌈Once the physical or virtual port can generate and validate MACsec traffic (signaled by `EthIf_MacSecOperational`), all Controllers using the virtual or physical port shall immediately communicate the MacSecOperational status to the upper layers with `EthSM_TrcvLinkStateChg`.⌋*(FO_RS_MACsec_00002)*

[SWS_EthIf_00583]{DRAFT} ⌈The Ethernet Interface module shall support the MKA related EtherTypes as defined in [23].⌋*(FO_RS_MACsec_00023)*

[SWS_EthIf_00584]{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 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_00587]{DRAFT} ⌈If `EthIfFwSupport` is set to FIREWALL_WITHOUT_PERSTREAMFILTERING, EthIf shall call `Fw_InspectPacket` for every network packet it receives. The `StreamHandleIdxPtrs` shall be set to NULL.⌋*()*

[SWS_EthIf_00588]{DRAFT} ⌈If `EthIfFwSupport` is set to FIREWALL_WITH_PERSTREAMFILTERING, EthIf shall call `EthSwt_ExtractStreamHandleIdx` to set the `StreamHandleIdxPtrs` before calling `Fw_InspectPacket`.⌋*()*

[SWS_EthIf_00589]{DRAFT} ⌈EthIf shall call `EthSwt_ExtractStreamHandleIdx` before carrying out the checks for raising SEvs specified in [SWS_EthIf_00503].⌋*()*

[SWS_EthIf_00590]{DRAFT} ⌈If `Fw_InspectPacket` is returned with `E_OK` and the `InspectionResultPtr` is set to FW_BLOCK_NETWORK_PACKET, the EthIf module shall drop the network packet without any further notice.⌋*()*

[SWS_EthIf_00591]{DRAFT} ⌈If `Fw_InspectPacket` is returned with `E_OK` and the `InspectionResultPtr` is set to FW_ALLOW_NETWORK_PACKET, the EthIf module shall forward the network packet to the next layer.⌋*()*

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]{DRAFT} ⌈If `EthIf_GetStreamHandleIdxStatistics` is called, the Ethif module shall call `EthSwt_GetStreamHandleIdxStatistics` with the same parameters. When the respective callback function `EthIf_StreamHandleIdxStatistics` is called, the EthIf module shall call `Fw_StreamHandleIdxStatistics` with the same parameters.⌋*()*

[SWS_EthIf_00593]{DRAFT} ⌈If `EthIf_SetStreamHandleIdxConfiguration` is called, the Ethif module shall call `EthSwt_SetStreamHandleIdxConfiguration` with the same parameters. When the respective callback function `EthIf_StreamHandleIdxConfiguration` is called, the EthIf module shall call `Fw_StreamHandleIdxConfiguration` with the same parameters.⌋*()*

7.2 Security Events

[SWS_EthIf_00502]{DRAFT} ⌈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].⌋*(RS_Ids_00810)*

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 ⌈

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

⌋*(RS_Ids_00810)*

Context data is not provided by the EthIf for the security events.

7.3 Error Classification

Section "Error Handling" of the document [6] "General Specification of Basic Software Modules" 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] Definiton 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
Invalid port index	ETHIF_E_INV_PORT_IDX	0x09

⌋*()*

7.3.2 Runtime Errors

[SWS_EthIf_91136] Definiton of runtime errors in module EthIf ⌈

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

⌋*(SRS_BSW_00385)*

7.3.3 Transient Faults

There are no transient faults.

7.3.4 Production Errors

There are no production errors.

7.3.5 Extended Production Errors

There are no extended production errors.

8 API specification

8.1 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*
ComStack_Types	ComStack_Types.h	BufReq_ReturnType
	ComStack_Types.h	ListElemStructType (draft)
	ComStack_Types.h	PduIdType
	ComStack_Types.h	PduInfoType
	ComStack_Types.h	PduLengthType
	ComStackTypes.h	TimeStampQualType (draft)
	ComStackTypes.h	TimeStampType (draft)
	ComStackTypes.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.h	Eth_SpiStatusType (draft)
	Eth_GeneralTypes.h	Eth_BufIdxType
	Eth_GeneralTypes.h	Eth_CounterType
	Eth_GeneralTypes.h	Eth_DataType
	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_TimeStampQualType (obsolete)
	Eth_GeneralTypes.h	Eth_TimeStampType (obsolete)
	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

▽

△

Module	Header File	Imported Type
	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
Fw	Fw_Types.h	Fw_InspectionResultType (draft)
IdsM	IdsM_Types.h	IdsM_SecurityEventIdType
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_GetChanRxParamIdType
	WEth_GeneralTypes.h	WEthTrcv_SetChanRxParamIdType
	WEth_GeneralTypes.h	WEthTrcv_SetChanTxParamIdType
	WEth_GeneralTypes.h	WEthTrcv_SetRadioParamIdType

⌋()

8.2 Type definitions

8.2.1 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

⌋()

8.2.2 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		

⌋()

8.2.3 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		

⌋()

8.2.4 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	

⌋()

8.3 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`

8.3.1 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`.⌋()

8.3.2 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 7.1.9.2 and 7.1.10.

[SWS_EthIf_00036] ⌈If development error detection is enabled: the function shall check that the service `EthIf_Init` was previously called. If the check fails, the function shall raise the development error `ETHIF_E_UNINIT`.⌋()

[SWS_EthIf_00037] ⌈If development error detection is enabled: the function shall check the parameter `CtrlIdx` for being valid. If the check fails, the function shall raise the development error `ETHIF_E_INV_CTRL_IDX`.⌋()

8.3.3 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).⌋()

[SWS_EthIf_00041] ⌈If development error detection is enabled: the function shall check that the service `EthIf_Init` was previously called. If the check fails, the function shall raise the development error `ETHIF_E_UNINIT`.⌋()

[SWS_EthIf_00042] ⌈If development error detection is enabled: the function shall check the parameter `CtrlIdx` for being valid. If the check fails, the function shall raise the development error `ETHIF_E_INV_CTRL_IDX`.⌋()

[SWS_EthIf_00043] ⌈If development error detection is enabled: the function shall check the parameter `CtrlModePtr` for being valid. If the check fails, the function shall raise the development error `ETHIF_E_PARAM_POINTER`.⌋()

8.3.4 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] ⌈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⌋ *(SRS_Eth_00106)*

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

[SWS_EthIf_00500] ⌈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.⌋ *(SRS_Eth_00106)*

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

[SWS_EthIf_00246] ⌈If development error detection is enabled: the function shall check that the service `EthIf_Init` was previously called. If the check fails, the function shall raise the development error `ETHIF_E_UNINIT`.⌋ *()*

[SWS_EthIf_00247] ⌈If development error detection is enabled: the function shall check the parameter `WakeupSource` for being valid. If the check fails, the function shall raise the development error `ETHIF_E_INV_PARAM`.⌋ *()*

8.3.5 EthIf_GetPhyWakeupReason

[SWS_EthIf_91004] Definition of API function EthIf_GetPhyWakeupReason ⌈

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	

⌋*(SRS_Eth_00107)*

[SWS_EthIf_00486] ⌈The function `EthIf_GetPhyWakeupReason` shall forward the call to function `EthTrcv_GetBusWuReason` of the corresponding Ethernet Transceiver Driver (EthIfTransceiverIdx).⌋*(SRS_Eth_00107)*

[SWS_EthIf_00487] ⌈If development error detection is enabled: the function shall check that the service `EthIf_Init` was previously called. If the check fails, the function shall raise the development error `ETHIF_E_UNINIT`.⌋*()*

[SWS_EthIf_00488] ⌈If development error detection is enabled: the function shall check the parameter `TrcvIdx` for being valid. If the check fails, the function shall raise the development error `ETHIF_E_INV_TRCV_IDX`.⌋*()*

[SWS_EthIf_00489] ⌈If development error detection is enabled: the function shall check the parameter `WakeupReasonPtr` for being valid. If the check fails, the function shall raise the development error `ETHIF_E_PARAM_POINTER`.⌋*()*

8.3.6 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] ⌈The function EthIf_GetSwitchPortWakeupReason shall forward the call to function EthSwt_GetSwitchPortWakeupReason of the corresponding Ethernet Switch Driver (EthIfSwitchIdx).⌋(*SRS_Eth_00107*)

[SWS_EthIf_00491] ⌈If development error detection is enabled: the function shall check that the service EthIf_Init was previously called. If the check fails, the function shall raise the development error ETHIF_E_UNINIT otherwise (if DET is disabled) return E_NOT_OK.⌋()

[SWS_EthIf_00492] ⌈If development error detection is enabled: the function shall check the parameter SwitchIdx for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_SWT_IDX otherwise (if DET is disabled) return E_NOT_OK.⌋()

[SWS_EthIf_00493] ⌈If development error detection is enabled: the function shall check the parameter SwitchPortIdx for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_PORT_IDX otherwise (if DET is disabled) return E_NOT_OK.⌋()

[SWS_EthIf_00494] ⌈If development error detection is enabled: the function shall check the parameter WakeupReasonPtr for being valid. If the check fails, the function shall raise the development error ETHIF_E_PARAM_POINTER.⌋()

8.3.7 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.⌋()

[SWS_EthIf_00063] ⌈If development error detection is enabled: the function shall check that the service `EthIf_Init` was previously called. If the check fails, the function shall raise the development error `ETHIF_E_UNINIT`.⌋()

[SWS_EthIf_00064] ⌈If development error detection is enabled: the function shall check the parameter `CtrlIdx` for being valid. If the check fails, the function shall raise the development error `ETHIF_E_INV_CTRL_IDX`.⌋()

[SWS_EthIf_00065] ⌈If development error detection is enabled: the function shall check the parameter `PhysAddrPtr` for being valid. If the check fails, the function shall raise the development error `ETHIF_E_PARAM_POINTER`.⌋()

8.3.8 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 Driver.
	PhysAddrPtr	Pointer to memory containing the physical source address (MAC address) in network byte order.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Sets the physical source address used by the indexed controller.	
Available via	EthIf.h	

⌋()

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

[SWS_EthIf_00135] ⌈If development error detection is enabled: the function shall check that the service EthIf_Init was previously called. If the check fails, the function shall raise the development error ETHIF_E_UNINIT.⌋*()*

[SWS_EthIf_00136] ⌈If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_CTRL_IDX.⌋*()*

[SWS_EthIf_00137] ⌈If development error detection is enabled: the function shall check the parameter PhysAddrPtr for being valid. If the check fails, the function shall raise the development error ETHIF_E_PARAM_POINTER.⌋*()*

8.3.9 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 Driver.
	PhysAddrPtr	Pointer to memory containing the physical destination address (MAC address) in network byte order. This is the multicast destination address of the layer 2 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, the software has to do this.	
Available via	EthIf.h	

⌋*()*

[SWS_EthIf_00140] ⌈The function EthIf_SetPhysAddrFilter shall forward the call to the respective Ethernet Controller Driver.⌋*()*

[SWS_EthIf_00141] ⌈If development error detection is enabled: the function shall check that the service EthIf_Init was previously called. If the check fails, the function shall raise the development error ETHIF_E_UNINIT.⌋*()*

[SWS_EthIf_00142] ⌈If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_CTRL_IDX.⌋()

[SWS_EthIf_00143] ⌈If development error detection is enabled: the function shall check the parameter PhysAddrPtr for being valid. If the check fails, the function shall raise the development error ETHIF_E_PARAM_POINTER.⌋()

8.3.10 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: EthIfGetPortMacAddrApi.⌋()

[SWS_EthIf_00193] ⌈If development error detection is enabled: the function shall check that the service EthIf_Init was previously called. If the check fails, the function shall raise the development error ETHIF_E_UNINIT.⌋()

[SWS_EthIf_00194] ⌈If development error detection is enabled: the function shall check the parameter MacAddrPtr, SwitchIdxPtr and PortIdxPtr for being valid. If the check fails, the function shall raise the development error ETHIF_E_PARAM_POINTER.⌋()

8.3.11 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`.⌋*()*

[SWS_EthIf_00199] ⌈If development error detection is enabled: the function shall check that the service `EthIf_Init` was previously called. If the check fails, the function shall raise the development error `ETHIF_E_UNINIT`.⌋*()*

[SWS_EthIf_00200] ⌈If development error detection is enabled: the function shall check the parameter `arlTableListPointer` for being valid. If the check fails, the function shall raise the development error `ETHIF_E_PARAM_POINTER`.⌋*()*

8.3.12 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.⌋()

[SWS_EthIf_00299] ⌈If development error detection is enabled: the function shall check that the service `EthIf_Init` was previously called. If the check fails, the function shall raise the development error `ETHIF_E_UNINIT`.⌋()

[SWS_EthIf_00300] ⌈If development error detection is enabled: the function shall check the OUT parameter `CtrlIdxListPtr` for being valid only if the the OUT parameter `NumberOfCtrlIdx` is greater 0x00. If the check fails, the function shall raise the development error `ETHIF_E_PARAM_POINTER`.⌋()

8.3.13 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`.⌋()

[SWS_EthIf_00302] ⌈If development error detection is enabled: the function shall check that the service `EthIf_Init` was previously called. If the check fails, the function shall raise the development error `ETHIF_E_UNINIT`.⌋()

[SWS_EthIf_00303] ⌈If development error detection is enabled: the function shall check the parameter `VlanIdPtr` for being valid. If the check fails, the function shall raise the development error `ETHIF_E_PARAM_POINTER`.⌋()

8.3.14 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.⌋()

[SWS_EthIf_00319] ⌈If development error detection is enabled: The function shall check that the service EthIf_Init was previously called. If the check fails, the function shall raise the development error ETHIF_E_NOTINIT.⌋()

8.3.15 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.⌋*()*

[SWS_EthIf_00217] ⌈If development error detection is enabled: the function shall check that the service EthIf_Init was previously called. If the check fails, the function shall raise the development error ETHIF_E_UNINIT.⌋*()*

8.3.16 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.⌋*()*

[SWS_EthIf_00222] ⌈If development error detection is enabled: the function shall check that the service EthIf_Init was previously called. If the check fails, the function shall raise the development error ETHIF_E_UNINIT.⌋*()*

8.3.17 EthIf_GetCurrentTime

[SWS_EthIf_00154]{OBSOLETE} **Definition of API function EthIf_GetCurrentTime** ⌈

Service Name	EthIf_GetCurrentTime (obsolete)	
Syntax	`Std_ReturnType EthIf_GetCurrentTime ( uint8 CtrlIdx, Eth_TimeStampQualType* timeQualPtr, Eth_TimeStampType* timeStampPtr )`	
Service ID [hex]	0x22	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	CtrlIdx	Index of the addresses ETH controller.
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: successful E_NOT_OK: failed
Description	Returns a time value out of the HW registers according to the capability of the HW. Is the HW resolution is lower than the Eth_TimeStampType resolution resp. range, the remaining bits will be filled with 0. Important Note: EthIf_GetCurrentTime may be called within an exclusive area. **Tags:** atp.Status=obsolete	
Available via	EthIf.h	

⌋()

[SWS_EthIf_00155]{OBSOLETE} ⌈If development error detection is enabled: the function shall check that the service EthIf_Init was previously called. If the check fails, the function shall raise the development error ETHIF_E_UNINIT.⌋()

[SWS_EthIf_00156]{OBSOLETE} ⌈If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_CTRL_IDX.⌋()

[SWS_EthIf_00157]{OBSOLETE} ⌈If development error detection is enabled: the function shall check the parameter timeQualPtr and timeStampPtr for being valid. If the check fails, the function shall raise the development error ETHIF_E_PARAM_POINTER.⌋()

[SWS_EthIf_00158]{OBSOLETE} ⌈The function shall be pre compile time configurable On/Off by the configuration parameter: EthIfGlobalTimeSupport.⌋()

[SWS_EthIf_00473]{OBSOLETE} ⌈The EthIf module shall apply appropriate mechanisms to allow calls of EthIf_GetCurrentTime API from other partitions than its main function, e.g. by providing an EthIf satellite.⌋()

8.3.18 EthIf_GetCurrentTimeTuple

[SWS_EthIf_91066]{DRAFT} **Definition of API function EthIf_GetCurrentTimeTuple** ⌈

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	

⌋*(SRS_Eth_00175)*

[SWS_EthIf_00601]{DRAFT} ⌈If development error detection is enabled: the function shall check that the service EthIf_Init was previously called. If the check fails, the function shall raise the development error ETHIF_E_UNINIT.⌋*(SRS_BSW_00386)*

[SWS_EthIf_00602]{DRAFT} ⌈If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_CTRL_IDX.⌋*(SRS_BSW_00386)*

[SWS_EthIf_00603]{DRAFT} ⌈If development error detection is enabled: the function shall check the parameter ClkUnitIdx for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_CLKUNIT_IDX.⌋*(SRS_BSW_00386)*

[SWS_EthIf_00604]{DRAFT} ⌈If development error detection is enabled: the function shall check the parameter currentTimeTuplePtr for being valid. If the check fails, the function shall raise the development error ETHIF_E_PARAM_POINTER.⌋*(SRS_BSW_00386)*

[SWS_EthIf_00585]{DRAFT} ⌈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 correspond-

ing EthIfPhysController and ClkUnitIdx to Ethernet controller clock unit which is referenced via EthIfClkUnitRef of the given ClkUnitIdx.⌋(*SRS_Eth_00175*)

[SWS_EthIf_00605]{DRAFT} ⌈The function shall be pre compile time configurable On/Off by the configuration parameter: EthIfGlobalTimeSupport.⌋(*SRS_BSW_00171*)

[SWS_EthIf_00606]{DRAFT} ⌈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.⌋(*SRS_BSW_00459*)

8.3.19 EthIf_SetPhcTime

[SWS_EthIf_91062]{DRAFT} **Definition of API function EthIf_SetPhcTime** ⌈

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	

⌋(*SRS_Eth_00175*)

[SWS_EthIf_00607]{DRAFT} ⌈If development error detection is enabled: the function shall check that the service EthIf_Init was previously called. If the check fails, the function shall raise the development error ETHIF_E_UNINIT.⌋(*SRS_BSW_00386*)

[SWS_EthIf_00608]{DRAFT} ⌈If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_CTRL_IDX.⌋(*SRS_BSW_00386*)

[SWS_EthIf_00609]{DRAFT} ⌈If development error detection is enabled: the function shall check the parameter ClkUnitIdx for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_CLKUNIT_IDX.⌋*(SRS_BSW_00386)*

[SWS_EthIf_00610]{DRAFT} ⌈The function shall be pre compile time configurable On/Off by the configuration parameter: `EthIfPhcSupport`.⌋*(SRS_BSW_00171)*

[SWS_EthIf_00586]{DRAFT} ⌈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`.⌋*(SRS_Eth_00175)*

[SWS_EthIf_00611]{DRAFT} ⌈The EthIf module shall apply appropriate mechanisms to allow calls of EthIf_SetPhcTime API from other partitions than its main function, e.g. by providing an EthIf satellite.⌋*(SRS_BSW_00459)*

8.3.20 EthIf_SetPhcCorrection

[SWS_EthIf_91063]{DRAFT} **Definition of API function EthIf_SetPhcCorrection** ⌈

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^{-41}), by which the PHC is requested to be corrected
	offset	Time offset, by which the PHC is requested to be updated.
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	

⌋*(SRS_Eth_00175)*

[SWS_EthIf_00620]{DRAFT} ⌈If development error detection is enabled: the function shall check that the service EthIf_Init was previously called. If the check fails, the function shall raise the development error ETHIF_E_UNINIT.⌋*(SRS_BSW_00386)*

[SWS_EthIf_00621]{DRAFT} ⌈If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_CTRL_IDX.⌋*(SRS_BSW_00386)*

[SWS_EthIf_00622]{DRAFT} ⌈If development error detection is enabled: the function shall check the parameter ClkUnitIdx for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_CLKUNIT_IDX.⌋*(SRS_BSW_00386)*

[SWS_EthIf_00623]{DRAFT} ⌈The function shall be pre compile time configurable On/Off by the configuration parameter: `EthIfPhcSupport`.⌋*(SRS_BSW_00171)*

[SWS_EthIf_00624]{DRAFT} ⌈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`.⌋*(SRS_Eth_00175)*

[SWS_EthIf_00625]{DRAFT} ⌈The EthIf module shall apply appropriate mechanisms to allow calls of EthIf_SetPhcTime API from other partitions than its main function, e.g. by providing an EthIf satellite.⌋*(SRS_BSW_00459)*

8.3.21 EthIf_GetPhcTime

[SWS_EthIf_91064]{DRAFT} **Definition of API function EthIf_GetPhcTime** ⌈

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	

⌋(*SRS_Eth_00175*)

[SWS_EthIf_00612]{DRAFT} ⌈If development error detection is enabled: the function shall check that the service EthIf_Init was previously called. If the check fails, the function shall raise the development error ETHIF_E_UNINIT.⌋(*SRS_BSW_00386*)

[SWS_EthIf_00626]{DRAFT} ⌈If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_CTRL_IDX.⌋(*SRS_BSW_00386*)

[SWS_EthIf_00627]{DRAFT} ⌈If development error detection is enabled: the function shall check the parameter ClkUnitIdx for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_CLKUNIT_IDX.⌋(*SRS_BSW_00386*)

[SWS_EthIf_00628]{DRAFT} ⌈If development error detection is enabled: the function shall check the parameter timeQualPtr and timeStampPtr for being valid. If the check fails, the function shall raise the development error ETHIF_E_PARAM_POINTER.⌋(*SRS_BSW_00386*)

[SWS_EthIf_00629]{DRAFT} ⌈If development error detection is enabled: the function shall check the parameter timeStampPtr and timeStampPtr for being valid. If the check fails, the function shall raise the development error ETHIF_E_PARAM_POINTER.⌋(*SRS_BSW_00386*)

[SWS_EthIf_00630]{DRAFT} ⌈The function shall be pre compile time configurable On/Off by the configuration parameter: `EthIfPhcSupport`.⌋(*SRS_BSW_00171*)

[SWS_EthIf_00631]{DRAFT} ⌈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`.⌋(*SRS_Eth_00175*)

[SWS_EthIf_00632]{DRAFT} ⌈The EthIf module shall apply appropriate mechanisms to allow calls of EthIf_GetPhcTime API from other partitions than its main function, e.g. by providing an EthIf satellite.⌋(*SRS_BSW_00459*)

8.3.22 EthIf_SetPpsSignalMode

[SWS_EthIf_91065]{DRAFT} **Definition of API function EthIf_SetPpsSignalMode** ⌈

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	

⌋*(SRS_Eth_00176)*

[SWS_EthIf_00633]{DRAFT} ⌈If development error detection is enabled: the function shall check that the service EthIf_Init was previously called. If the check fails, the function shall raise the development error ETHIF_E_UNINIT.⌋*(SRS_BSW_00386)*

[SWS_EthIf_00634]{DRAFT} ⌈If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_CTRL_IDX.⌋*(SRS_BSW_00386)*

[SWS_EthIf_00635]{DRAFT} ⌈The function shall be pre compile time configurable On/ Off by the configuration parameter: `EthIfPhcSupport`.⌋*(SRS_BSW_00171)*

[SWS_EthIf_00636]{DRAFT} ⌈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`.⌋*(SRS_Eth_00175)*

8.3.23 EthIf_EnableEgressTimeStamp

[SWS_EthIf_00160] Definition of API function EthIf_EnableEgressTimeStamp ⌈

Service Name	EthIf_EnableEgressTimeStamp	

Syntax	`void EthIf_EnableEgressTimeStamp (`
	`  uint8 CtrlIdx,`
	`  Eth_BufIdxType BufIdx`
	`)`	

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

⌋()

[SWS_EthIf_00161] ⌈If development error detection is enabled: the function shall check that the service `EthIf_Init` was previously called. If the check fails, the function shall raise the development error `ETHIF_E_UNINIT`.⌋()

[SWS_EthIf_00162] ⌈If development error detection is enabled: the function shall check the parameter `CtrlIdx` for being valid. If the check fails, the function shall raise the development error `ETHIF_E_INV_CTRL_IDX`.⌋()

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

8.3.24 EthIf_GetEgressTimeStamp

[SWS_EthIf_00166] Definition of API function EthIf_GetEgressTimeStamp ⌈

Service Name	EthIf_GetEgressTimeStamp

Syntax	`Std_ReturnType EthIf_GetEgressTimeStamp (`
	`  uint8 CtrlIdx,`
	`  Eth_BufIdxType BufIdx,`
	`  Eth_TimeStampQualType* timeQualPtr,`
	`  Eth_TimeStampType* timeStampPtr`
	`)`

Service ID [hex]	0x24
Sync/Async	Synchronous

▽

△

Reentrancy	Non Reentrant	
Parameters (in)	CtrlIdx	Index of the address ETH controller.
	BufIdx	Index of the message buffer, where the Upper Layer expects egress time stamping
Parameters (inout)	None	
Parameters (out)	timeQualPtr	quality of HW time stamp, e.g. based on current drift
	timeStampPtr	current time stamp
Return value	Std_ReturnType	E_OK: success 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_00167] ⌈If development error detection is enabled: the function shall check that the service EthIf_Init was previously called. If the check fails, the function shall raise the development error ETHIF_E_UNINIT.⌋()

[SWS_EthIf_00168] ⌈If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_CTRL_IDX.⌋()

[SWS_EthIf_00169] ⌈If development error detection is enabled: the function shall check the parameter timeQualPtr and timeStampPtr for being valid. If the check fails, the function shall raise the development error ETHIF_E_PARAM_POINTER.⌋()

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

8.3.25 EthIf_GetIngressTimeStamp

[SWS_EthIf_00172] Definition of API function EthIf_GetIngressTimeStamp ⌈

Service Name	EthIf_GetIngressTimeStamp	

Syntax	`Std_ReturnType EthIf_GetIngressTimeStamp (`
	`  uint8 CtrlIdx,`
	`  const Eth_DataType* DataPtr,`
	`  Eth_TimeStampQualType* timeQualPtr,`
	`  Eth_TimeStampType* timeStampPtr`
	`)`	

Service ID [hex]	0x25	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	CtrlIdx	Index of the addresses ETH controller.
	DataPtr	Pointer to the message buffer, where Application expects ingress time stamping
Parameters (inout)	None	

▽

△

Parameters (out)	timeQualPtr	quality of HW time stamp, e.g. based on current drift
	timeStampPtr	current time stamp
Return value	Std_ReturnType	E_OK: success 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.	
Available via	EthIf.h	

⌋()

[SWS_EthIf_00173] ⌈If development error detection is enabled: the function shall check that the service EthIf_Init was previously called. If the check fails, the function shall raise the development error ETHIF_E_UNINIT.⌋()

[SWS_EthIf_00174] ⌈If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_CTRL_IDX.⌋()

[SWS_EthIf_00175] ⌈If development error detection is enabled: the function shall check the parameter DataPtr, timeQualPtr and timeStampPtr for being valid. If the check fails, the function shall raise the development error ETHIF_E_PARAM_POINTER.⌋()

[SWS_EthIf_00176] ⌈The function shall be pre compile time configurable On/Off by the configuration parameter: EthIfGlobalTimeSupport.⌋()

8.3.26 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.

[SWS_EthIf_00273] ⌈If development error detection is enabled: the function shall check that the service EthIf_Init was previously called. If the check fails, the function shall raise the development error ETHIF_E_UNINIT.⌋()

[SWS_EthIf_00274] ⌈If development error detection is enabled: the function shall check that the provided parameter PortGroupIdx addresses a port group not referenced by any EthIfController. If the check fails, the function shall raise the development error ETHIF_E_INV_PORT_GROUP_IDX.⌋()

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

8.3.27 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	

⌋()

[SWS_EthIf_00277] ⌈If development error detection is enabled: the function shall check that the service `EthIf_Init` was previously called. If the check fails, the function shall raise the development error `ETHIF_E_UNINIT`.⌋()

8.3.28 EthIf_SetSwitchMgmtInfo

[SWS_EthIf_91003] Definition of API function EthIf_SetSwitchMgmtInfo ⌈

Service Name	EthIf_SetSwitchMgmtInfo	
Syntax	`Std_ReturnType EthIf_SetSwitchMgmtInfo ( uint8 CtrlIdx, Eth_BufIdxType BufIdx, EthSwt_MgmtInfoType* MgmtInfoPtr )`	
Service ID [hex]	0x38	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	CtrlIdx	Index of an Ethernet Interface controller
	BufIdx	Ethernet Tx Buffer index
	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. It has to be called between EthIf_ProvideTxBuffer() and EthIf_Transmit() of the related frame.	
Available via	EthIf.h	

⌋(*SRS_Eth_00125*)

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

[SWS_EthIf_00280] ⌈If development error detection is enabled: the function shall check that the service EthIf_Init() was previously called. If the check fails, the function shall raise the development error ETHIF_E_UNINIT.⌋()

[SWS_EthIf_00281] ⌈If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_CTRL_IDX.⌋()

[SWS_EthIf_00282] ⌈If development error detection is enabled: the function shall check the parameter BufIdx for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_PARAM.⌋()

[SWS_EthIf_00283] ⌈If development error detection is enabled: the function shall check the parameter MgmtInfoPtr for being valid. If the check fails, the function shall raise the development error ETHIF_E_PARAM_POINTER.⌋()

8.3.29 EthIf_GetRxMgmtObject

[SWS_EthIf_91105] Definition of API function EthIf_GetRxMgmtObject ⌈

Service Name	EthIf_GetRxMgmtObject	
Syntax	`Std_ReturnType EthIf_GetRxMgmtObject ( uint8 CtrlIdx, Eth_DataType* DataPtr, EthSwt_MgmtObjectType **MgmtObjectPtr )`	
Service ID [hex]	0x47	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	CtrlIdx	Index of an Ethernet Interface controller
	DataPtr	Ethernet data pointer
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	

⌋()

8.3.30 EthIf_GetTxMgmtObject

[SWS_EthIf_91106] Definition of API function EthIf_GetTxMgmtObject ⌈

Service Name	EthIf_GetTxMgmtObject	
Syntax	`Std_ReturnType EthIf_GetTxMgmtObject ( uint8 CtrlIdx, Eth_BufIdxType BufIdx, EthSwt_MgmtObjectType **MgmtObjectPtr )`	
Service ID [hex]	0x48	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	CtrlIdx	Index of an Ethernet Interface controller
	BufIdx	Ethernet Rx Buffer index
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	

⌋()

8.3.31 EthIf_SwitchEnableTimeStamping

[SWS_EthIf_91007] Definition of API function EthIf_SwitchEnableTimeStamping ⌈

Service Name	EthIf_SwitchEnableTimeStamping	
Syntax	`Std_ReturnType EthIf_SwitchEnableTimeStamping ( uint8 CtrlIdx, Eth_BufIdxType BufIdx, EthSwt_MgmtInfoType* MgmtInfo )`	
Service ID [hex]	0x39	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	CtrlIdx	Index of the Ethernet controller within the context of the Ethernet Interface
	BufIdx	Index of the message buffer, where Application expects egress time stamping
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 by CtrlIdx and Buf Idx.	
Available via	EthIf.h	

⌋*(SRS_Eth_00125)*

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

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

[SWS_EthIf_00286] ⌈If development error detection is enabled: the function shall check that the service `Eth_Init` was previously called. If the check fails, the function shall raise the development error `ETHIF_E_UNINIT`.⌋*()*

[SWS_EthIf_00287] ⌈If development error detection is enabled: the function shall check the parameter `CtrlIdx` for being valid. If the check fails, the function shall raise the development error `ETHIF_E_INV_CTRL_IDX`.⌋*()*

[SWS_EthIf_00288] ⌈If development error detection is enabled: the function shall check the parameter `BufIdx` for being valid. If the check fails, the function shall raise the development error `ETHIF_E_INV_PARAM`.⌋*()*

[SWS_EthIf_00289] ⌈If development error detection is enabled: the function shall check the parameter `BufIdx` for being valid. If the check fails, the function shall raise the development error `ETHIF_E_INV_PARAM`.⌋*()*

[SWS_EthIf_00290] ⌈If development error detection is enabled: the function shall check the parameter `BufIdx` for being valid. If the check fails, the function shall raise the development error `ETHIF_E_INV_PARAM`.⌋*()*

8.3.32 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_00304] ⌈If development error detection is enabled: the function shall check that the service `EthIf_Init` was previously called. If the check fails, the function shall raise the development error `ETHIF_E_UNINIT`.⌋*(SRS_BSW_00101, SRS_BSW_00369)*

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

8.3.33 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_00306] ⌈If development error detection is enabled: the function shall check that the service `EthIf_Init` was previously called. If the check fails, the function shall raise the development error `ETHIF_E_UNINIT`.⌋*(SRS_BSW_00101, SRS_BSW_00369)*

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

8.3.34 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 (EthIfTransceiverIdx).⌋()

[SWS_EthIf_00392] ⌈If development error detection is enabled: the function shall check that the service EthIf_Init was previously called. If the check fails, the function shall raise the development error ETHIF_E_UNINIT.⌋()

[SWS_EthIf_00393] ⌈If development error detection is enabled: the function shall check the parameter TrcvIdx for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_TRCV_IDX.⌋()

[SWS_EthIf_00394] ⌈If development error detection is enabled: the function shall check the parameter ResultPtr for being valid. If the check fails, the function shall raise the development error ETHIF_E_PARAM_POINTER.⌋()

8.3.35 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).⌋()

[SWS_EthIf_00396] ⌈If development error detection is enabled: the function shall check that the service EthIf_Init was previously called. If the check fails, the function shall raise the development error ETHIF_E_UNINIT.⌋()

[SWS_EthIf_00397] ⌈If development error detection is enabled: the function shall check the parameter SwitchIdx for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_SWT_IDX.⌋()

[SWS_EthIf_00495] ⌈If development error detection is enabled: the function shall check the parameter SwitchPortIdx for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_PORT_IDX otherwise (if DET is disabled) return E_NOT_OK.⌋()

[SWS_EthIf_00399] ⌈If development error detection is enabled: the function shall check the parameter ResultPtr for being valid. If the check fails, the function shall raise the development error ETHIF_E_PARAM_POINTER.⌋()

8.3.36 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`.⌋()

[SWS_EthIf_00401] ⌈If development error detection is enabled: the function shall check that the service `EthIf_Init` was previously called. If the check fails, the function shall raise the development error `ETHIF_E_UNINIT`.⌋()

[SWS_EthIf_00402] ⌈If development error detection is enabled: the function shall check the parameter `TrcvIdx` for being valid. If the check fails, the function shall raise the development error `ETHIF_E_INV_TRCV_IDX`.⌋()

8.3.37 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.⌋()

[SWS_EthIf_00405] ⌈If development error detection is enabled: the function shall check that the service EthIf_Init was previously called. If the check fails, the function shall raise the development error ETHIF_E_UNINIT.⌋()

[SWS_EthIf_00406] ⌈If development error detection is enabled: the function shall check the parameter SwitchIdx for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_SWT_IDX.⌋()

[SWS_EthIf_00496] ⌈If development error detection is enabled: the function shall check the parameter SwitchPortIdx for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_PORT_IDX otherwise (if DET is disabled) return E_NOT_OK.⌋()

8.3.38 EthIf_SetPhyTestMode

[SWS_EthIf_91016] Definition of API function EthIf_SetPhyTestMode ⌈

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	

⌋(*SRS_Eth_00117*)

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

[SWS_EthIf_00325] ⌈If development error detection is enabled: the function shall check that the service EthIf_Init was previously called. If the check fails, the function shall raise the development error ETHIF_E_UNINIT.⌋*()*

[SWS_EthIf_00326] ⌈If development error detection is enabled: the function shall check the parameter TrcvIdx for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_TRCV_IDX.⌋*()*

8.3.39 EthIf_SetPhyLoopbackMode

[SWS_EthIf_91018] Definition of API function EthIf_SetPhyLoopbackMode ⌈

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	

⌋(*SRS_Eth_00117*)

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

[SWS_EthIf_00328] ⌈If development error detection is enabled: the function shall check that the service EthIf_Init was previously called. If the check fails, the function shall raise the development error ETHIF_E_UNINIT.⌋*()*

[SWS_EthIf_00329] ⌈If development error detection is enabled: the function shall check the parameter `TrcvIdx` for being valid. If the check fails, the function shall raise the development error `ETHIF_E_INV_TRCV_IDX`.⌋*()*

8.3.40 EthIf_SetPhyTxMode

[SWS_EthIf_91061] Definition of API function EthIf_SetPhyTxMode ⌈

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	

⌋*(SRS_Eth_00117)*

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

[SWS_EthIf_00389] ⌈If development error detection is enabled: the function shall check that the service `EthIf_Init` was previously called. If the check fails, the function shall raise the development error `ETHIF_E_UNINIT`.⌋*()*

[SWS_EthIf_00390] ⌈If development error detection is enabled: the function shall check the parameter `TrcvIdx` for being valid. If the check fails, the function shall raise the development error `ETHIF_E_INV_TRCV_IDX`.⌋*()*

8.3.41 EthIf_GetCableDiagnosticsResult

[SWS_EthIf_91014] Definition of API function EthIf_GetCableDiagnosticsResult ⌈

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	

⌋*(SRS_Eth_00117)*

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

[SWS_EthIf_00331] ⌈If development error detection is enabled: the function shall check that the service `EthIf_Init` was previously called. If the check fails, the function shall raise the development error `ETHIF_E_UNINIT`.⌋*()*

[SWS_EthIf_00332] ⌈If development error detection is enabled: the function shall check the parameter `TrcvIdx` for being valid. If the check fails, the function shall raise the development error `ETHIF_E_INV_TRCV_IDX`.⌋*()*

[SWS_EthIf_00333] ⌈If development error detection is enabled: the function shall check the parameter `ResultPtr` for being valid. If the check fails, the function shall raise the development error `ETHIF_E_PARAM_POINTER`.⌋*()*

8.3.42 EthIf_GetPhyIdentifier

[SWS_EthIf_91020] Definition of API function EthIf_GetPhyIdentifier ⌈

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	

⌋*(SRS_Eth_00117)*

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

[SWS_EthIf_00335] ⌈If development error detection is enabled: the function shall check that the service `EthIf_Init` was previously called. If the check fails, the function shall raise the development error `ETHIF_E_UNINIT`.⌋*()*

[SWS_EthIf_00336] ⌈If development error detection is enabled: the function shall check the parameter `TrcvIdx` for being valid. If the check fails, the function shall raise the development error `ETHIF_E_INV_TRCV_IDX`.⌋*()*

[SWS_EthIf_00337] ⌈If development error detection is enabled: the function shall check the parameter `OrgUniqueIdPtr` for being valid. If the check fails, the function shall raise the development error `ETHIF_E_PARAM_POINTER`.⌋*()*

[SWS_EthIf_00338] ⌈If development error detection is enabled: the function shall check the parameter `ModelNrPtr` for being valid. If the check fails, the function shall raise the development error `ETHIF_E_PARAM_POINTER`.⌋*()*

[SWS_EthIf_00339] ⌈If development error detection is enabled: the function shall check the parameter `RevisionNrPtr` for being valid. If the check fails, the function shall raise the development error `ETHIF_E_PARAM_POINTER`.⌋*()*

8.3.43 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`.⌋()

[SWS_EthIf_00343] ⌈If development error detection is enabled: the function shall check that the service `EthIf_Init` was previously called. If the check fails, the function shall raise the development error `ETHIF_E_UNINIT`.⌋()

[SWS_EthIf_00344] ⌈If development error detection is enabled: the function shall check the parameter `CtrlIdx` for being valid. If the check fails, the function shall raise the development error `ETHIF_E_INV_CTRL_IDX`.⌋()

[SWS_EthIf_00345] ⌈If development error detection is enabled: the function shall check the parameter `RxParamIds` for being valid. If the check fails, the function shall raise the development error `ETHIF_E_PARAM_POINTER`.⌋()

[SWS_EthIf_00346] ⌈If development error detection is enabled: the function shall check the parameter `ParamValues` for being valid. If the check fails, the function shall raise the development error `ETHIF_E_PARAM_POINTER`.⌋()

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

8.3.44 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`.⌋()

[SWS_EthIf_00349] ⌈If development error detection is enabled: the function shall check that the service `EthIf_Init` was previously called. If the check fails, the function shall raise the development error `ETHIF_E_UNINIT`.⌋()

[SWS_EthIf_00350] ⌈If development error detection is enabled: the function shall check the parameter `CtrlIdx` for being valid. If the check fails, the function shall raise the development error `ETHIF_E_INV_CTRL_IDX`.⌋()

[SWS_EthIf_00351] ⌈If development error detection is enabled: the function shall check the parameter `TxParamIds` for being valid. If the check fails, the function shall raise the development error `ETHIF_E_PARAM_POINTER`.⌋()

[SWS_EthIf_00352] ⌈If development error detection is enabled: the function shall check the parameter `ParamValues` for being valid. If the check fails, the function shall raise the development error `ETHIF_E_PARAM_POINTER`.⌋()

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

8.3.45 EthIf_SetBufWTxParams

[SWS_EthIf_91017] Definition of API function EthIf_SetBufWTxParams ⌈

Service Name	EthIf_SetBufWTxParams	

Syntax	`Std_ReturnType EthIf_SetBufWTxParams (`
	`  uint8 CtrlIdx,`
	`  Eth_BufIdxType BufIdx,`
	`  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
	BufIdx	Index of the buffer resource
	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`.⌋()

[SWS_EthIf_00355] ⌈If development error detection is enabled: the function shall check that the service `EthIf_Init` was previously called. If the check fails, the function shall raise the development error `ETHIF_E_UNINIT`.⌋()

[SWS_EthIf_00356] ⌈If development error detection is enabled: the function shall check the parameter `CtrlIdx` for being valid. If the check fails, the function shall raise the development error `ETHIF_E_INV_CTRL_IDX`.⌋()

[SWS_EthIf_00357] ⌈If development error detection is enabled: the function shall check the parameter `BufIdx` for being valid. If the check fails, the function shall raise the development error `ETHIF_E_INV_PARAM`.⌋()

[SWS_EthIf_00358] ⌈If development error detection is enabled: the function shall check the parameter `TxParamIds` for being valid. If the check fails, the function shall raise the development error `ETHIF_E_PARAM_POINTER`.⌋()

[SWS_EthIf_00359] ⌈If development error detection is enabled: the function shall check the parameter ParamValues for being valid. If the check fails, the function shall raise the development error ETHIF_E_PARAM_POINTER.⌋*()*

Note: The function requires previous buffer request (EthIf_ProvideTxBuffer).

8.3.46 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.⌋*()*

[SWS_EthIf_00362] ⌈If development error detection is enabled: the function shall check that the service EthIf_Init was previously called. If the check fails, the function shall raise the development error ETHIF_E_UNINIT.⌋*()*

[SWS_EthIf_00363] ⌈If development error detection is enabled: the function shall check the parameter TrcvId for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_TRCV_IDX.⌋*()*

[SWS_EthIf_00364] ⌈If development error detection is enabled: the function shall check the parameter ParamIds for being valid. If the check fails, the function shall raise the development error ETHIF_E_PARAM_POINTER.⌋*()*

[SWS_EthIf_00365] ⌈If development error detection is enabled: the function shall check the parameter ParamValue for being valid. If the check fails, the function shall raise the development error ETHIF_E_PARAM_POINTER.⌋*()*

8.3.47 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.⌋*()*

[SWS_EthIf_00368] ⌈If development error detection is enabled: the function shall check that the service EthIf_Init was previously called. If the check fails, the function shall raise the development error ETHIF_E_UNINIT.⌋*()*

[SWS_EthIf_00369] ⌈If development error detection is enabled: the function shall check the parameter TrcvId for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_TRCV_IDX.⌋*()*

[SWS_EthIf_00370] ⌈If development error detection is enabled: the function shall check the parameter RadioId for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_PARAM.⌋*()*

[SWS_EthIf_00371] ⌈If development error detection is enabled: the function shall check the parameter `ParamIds` for being valid. If the check fails, the function shall raise the development error `ETHIF_E_PARAM_POINTER`.⌋*()*

[SWS_EthIf_00372] ⌈If development error detection is enabled: the function shall check the parameter `ParamValues` for being valid. If the check fails, the function shall raise the development error `ETHIF_E_PARAM_POINTER`.⌋*()*

8.3.48 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`.⌋*()*

[SWS_EthIf_00375] ⌈If development error detection is enabled: the function shall check that the service `EthIf_Init` was previously called. If the check fails, the function shall raise the development error `ETHIF_E_UNINIT`.⌋*()*

[SWS_EthIf_00376] ⌈If development error detection is enabled: the function shall check the parameter `TrcvId` for being valid. If the check fails, the function shall raise the development error `ETHIF_E_INV_TRCV_IDX`.⌋*()*

[SWS_EthIf_00377] ⌈If development error detection is enabled: the function shall check the parameter RadioId for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_PARAM.⌋()

[SWS_EthIf_00378] ⌈If development error detection is enabled: the function shall check the parameter TxParamIds for being valid. If the check fails, the function shall raise the development error ETHIF_E_PARAM_POINTER.⌋()

[SWS_EthIf_00379] ⌈If development error detection is enabled: the function shall check the parameter ParamValues for being valid. If the check fails, the function shall raise the development error ETHIF_E_PARAM_POINTER.⌋()

8.3.49 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.⌋()

[SWS_EthIf_00382] ⌈If development error detection is enabled: the function shall check that the service EthIf_Init was previously called. If the check fails, the function shall raise the development error ETHIF_E_UNINIT.⌋()

[SWS_EthIf_00383] ⌈If development error detection is enabled: the function shall check the parameter TrcvId for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_TRCV_IDX.⌋()

[SWS_EthIf_00384] ⌈If development error detection is enabled: the function shall check the parameter RadioId for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_PARAM.⌋()

[SWS_EthIf_00385] ⌈If development error detection is enabled: the function shall check the parameter ParamIds for being valid. If the check fails, the function shall raise the development error ETHIF_E_PARAM_POINTER.⌋()

[SWS_EthIf_00386] ⌈If development error detection is enabled: the function shall check the parameter ParamValues for being valid. If the check fails, the function shall raise the development error ETHIF_E_PARAM_POINTER.⌋()

8.3.50 EthIf_ProvideTxBuffer

[SWS_EthIf_00067] Definition of API function EthIf_ProvideTxBuffer ⌈

Service Name	EthIf_ProvideTxBuffer	
Syntax	`BufReq_ReturnType EthIf_ProvideTxBuffer ( uint8 CtrlIdx, Eth_FrameType FrameType, uint8 Priority, Eth_BufIdxType* BufIdxPtr, uint8** BufPtr, uint16* LenBytePtr )`	
Service ID [hex]	0x09	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	CtrlIdx	Index of the Ethernet controller within the context of the Ethernet Interface
	FrameType	Ethernet Frame Type (EtherType)
	Priority	Priority value which shall be used for the 3-bit PCP field of the VLAN tag
Parameters (inout)	LenBytePtr	in: desired length in bytes, out: granted length in bytes
Parameters (out)	BufIdxPtr	Index to the granted buffer resource. To be used for subsequent requests
	BufPtr	Pointer to the granted buffer
Return value	BufReq_ReturnType	BUFREQ_OK: success BUFREQ_E_NOT_OK: development error detected BUFREQ_E_BUSY: all buffers in use BUFREQ_E_OVFL: requested buffer too large
Description	Provides access to a transmit buffer of the specified Ethernet controller.	
Available via	EthIf.h	

⌋()

[SWS_EthIf_00146] ⌈If CtrlIdx refers to an EthIfCtrl where no EthIfVlanID is configured, the parameters FrameType and Priority are not used.⌋()

[SWS_EthIf_00147] ⌈If VLAN is used

- EthIf shall increment the input desired length by 4 bytes before calling the Ethernet Driver module
- EthIf shall store the PCP (`Priority` parameter), CFI (always 0), VID (configured VLAN ID) and value of the `FrameType` parameter at the beginning of the buffer received from `<EthDrv>_ProvideTxBuffer`).
- EthIf shall increment the `BufPtr` by 4 bytes when returning the granted buffer
- EthIf shall decrement the output granted length by 4 bytes

⌋()

[SWS_EthIf_00068] ⌈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_ProvideTxBuffer` shall forward the call to the respective Ethernet Controller Driver or CanXL Controller Driver. Otherwise the function shall reject the request for a transmission buffer and return with `E_NOT_OK`.⌋()

[SWS_EthIf_00069] ⌈If development error detection is enabled: the function shall check that the service `EthIf_Init` was previously called. If the check fails, the function shall raise the development error `ETHIF_E_UNINIT`.⌋()

[SWS_EthIf_00070] ⌈If development error detection is enabled: the function shall check the parameter `CtrlIdx` for being valid. If the check fails, the function shall raise the development error `ETHIF_E_INV_CTRL_IDX`.⌋()

[SWS_EthIf_00071] ⌈If development error detection is enabled: the function shall check the parameter `BufIdxPtr` for being valid. If the check fails, the function shall raise the development error `ETHIF_E_PARAM_POINTER`.⌋()

[SWS_EthIf_00072] ⌈If development error detection is enabled: the function shall check the parameter `BufPtr` for being valid. If the check fails, the function shall raise the development error `ETHIF_E_PARAM_POINTER`.⌋()

[SWS_EthIf_00073] ⌈If development error detection is enabled: the function shall check the parameter `LenBytePtr` for being valid. If the check fails, the function shall raise the development error `ETHIF_E_PARAM_POINTER`.⌋()

8.3.51 EthIf_Transmit

[SWS_EthIf_00075] Definition of API function EthIf_Transmit ⌈

Service Name	EthIf_Transmit	
Syntax	`Std_ReturnType EthIf_Transmit ( uint8 CtrlIdx, Eth_BufIdxType BufIdx, Eth_FrameType FrameType, boolean TxConfirmation, uint16 LenByte, const uint8* PhysAddrPtr )`	
Service ID [hex]	0x0a	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different buffer indexes and Ctrl indexes	
Parameters (in)	CtrlIdx	Index of the Ethernet controller within the context of the Ethernet Interface
	BufIdx	Index of the buffer resource
	FrameType	Ethernet frame type
	TxConfirmation	Activates transmission confirmation
	LenByte	Data length in byte
	PhysAddrPtr	Physical target address (MAC address) in network byte order
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: success E_NOT_OK: transmission failed
Description	Triggers transmission of a previously filled transmit buffer	
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.⌋()

[SWS_EthIf_00077] ⌈If development error detection is enabled: the function shall check that the service EthIf_Init was previously called. If the check fails, the function shall raise the development error ETHIF_E_UNINIT.⌋()

[SWS_EthIf_00078] ⌈If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_CTRL_IDX.⌋()

[SWS_EthIf_00079] ⌈If development error detection is enabled: the function shall check the parameter BufIdx for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_PARAM.⌋()

[SWS_EthIf_00080] ⌈If development error detection is enabled: the function shall check the parameter PhysAddrPtr for being valid. If the check fails, the function shall raise the development error ETHIF_E_PARAM_POINTER.⌋*()*

8.3.52 EthIf_ImmediateTransmit

[SWS_EthIf_91137]{DRAFT} **Definition of API function EthIf_ImmediateTransmit** ⌈

Service Name	EthIf_ImmediateTransmit (draft)	
Syntax	`Std_ReturnType EthIf_ImmediateTransmit ( PduIdType TxPduId, const PduInfoType* PduInfoPtr )`	
Service ID [hex]	0x9a	
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 been rejected.
Description	Request transmission of a PDU. **Tags:** atp.Status=draft	
Available via	EthIf.h	

⌋*(SRS_Eth_00169)*

[SWS_EthIf_00614]{DRAFT} ⌈If development error detection is enabled: the function shall check that the service EthIf_Init was previously called. If the check fails, the function shall raise the development error ETHIF_E_UNINIT.⌋*()*

[SWS_EthIf_00615]{DRAFT} ⌈If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_CTRL_IDX.⌋*()*

[SWS_EthIf_00616]{DRAFT} ⌈If development error detection is enabled: the function shall check the parameter DataPtr for being valid. If the check fails, the function shall raise the development error ETHIF_E_PARAM_POINTER.⌋*()*

8.3.53 EthIf_ReleaseRxBuffer

[SWS_EthIf_91138]{DRAFT} **Definition of API function EthIf_ReleaseRxBuffer** ⌈

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	

⌋(*SRS_Eth_00169*)

[SWS_EthIf_00617]{DRAFT} ⌈If development error detection is enabled: the function shall check that the service `EthIf_Init` was previously called. If the check fails, the function shall raise the development error `ETHIF_E_UNINIT`.⌋()

[SWS_EthIf_00618]{DRAFT} ⌈If development error detection is enabled: the function shall check the parameter `CtrlIdx` for being valid. If the check fails, the function shall raise the development error `ETHIF_E_INV_CTRL_IDX`.⌋()

[SWS_EthIf_00619]{DRAFT} ⌈If development error detection is enabled: the function shall check the parameter `DataPtr` for being valid. If the check fails, the function shall raise the development error `ETHIF_E_PARAM_POINTER`.⌋()

8.3.54 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	

⌋()

[SWS_EthIf_00127] ⌈If development error detection is enabled: the function shall check the parameter `VersionInfoPtr` for being valid. If the check fails, the function shall raise the development error `ETHIF_E_PARAM_POINTER`.⌋*()*

8.3.55 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).⌋*()*

8.3.56 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 Transceiver Driver
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 (EthIfTransceiverIdx).⌋()

8.3.57 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).⌋()

8.3.58 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 Transceiver Driver
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 (EthIfTransceiverIdx).⌋()

8.3.59 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).⌋()

8.3.60 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 Transceiver Driver
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 (EthIfTransceiverIdx).⌋()

8.3.61 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).⌋()

8.3.62 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 Transceiver Driver
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 (EthIfTransceiverIdx).⌋()

8.3.63 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).⌋()

8.3.64 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).⌋()

8.3.65 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).⌋()

8.3.66 EthIf_SwitchPortGetTxErrorCounterValues

[SWS_EthIf_91118] Definition of API function EthIf_SwitchPortGetTxError CounterValues ⌈

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_SwitchPortGetTxErrorCounterVal-ues shall forward the call to function EthSwt_GetTxErrorCounterValues of the corresponding Ethernet Switch Driver (EthIfSwitchIdx).⌋()

8.3.67 EthIf_SwitchPortGetMacLearningMode

[SWS_EthIf_91119] Definition of API function EthIf_SwitchPortGetMacLearning Mode ⌈

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).⌋()

8.3.68 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).⌋*()*

8.3.69 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).⌋*()*

8.3.70 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).⌋()

8.3.71 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).⌋()

8.3.72 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).⌋()

8.3.73 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).⌋()

8.3.74 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).⌋()

8.3.75 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).⌋()

8.3.76 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).⌋()

8.3.77 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).⌋()

8.3.78 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).⌋()

8.3.79 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).⌋()

8.3.80 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 Transceiver Driver.
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 (EthIfTransceiverIdx).⌋()

8.3.81 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).⌋()

8.3.82 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).⌋()

8.3.83 EthIf_SwitchPortGetMaxFIFOBufferFillLevel

[SWS_EthIf_91135] Definition of API function EthIf_SwitchPortGetMaxFIFOBufferFillLevel ⌈

Service Name	EthIf_SwitchPortGetMaxFIFOBufferFillLevel	
Syntax	`Std_ReturnType EthIf_SwitchPortGetMaxFIFOBufferFillLevel ( uint8 SwitchPortIdx, uint8 PortIdx, uint8 SwitchPortEgressFifoIdx, uint32* SwitchPortEgressFifoBufferLevelPtr )`	
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.
	SwitchPortEgressFifoIdx	Index of the egress FIFO of the addressed Ethernet switch port
Parameters (inout)	None	
Parameters (out)	SwitchPortEgressFifo BufferLevelPtr	Pointer to a memory location, where the maximum amount of allocated FIFO buffer (in bytes) since the last read out shall be stored
Return value	Std_ReturnType	E_OK: success E_NOT_OK: The maximal FIFO buffer level could not be obtained
Description	The function retrieves the maximum amount of allocated FIFO buffer of the indexed Ethernet switch egress port. If the Ethernet switch hardware does not support Ethernet switch port based maximal FIFO buffer level, the content of SwitchPortEgressFifoBufferLevelPtr 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_SwitchPortGetMaxFIFOBufferFillLevel shall forward the call to function EthSwt_GetMaxFIFOBufferFillLevel of the corresponding Ethernet Switch Driver (EthIfSwitchIdx).⌋()

8.3.84 EthIf_TransceiverGetMacMethod

[SWS_EthIf_91021] Definition of API function EthIf_TransceiverGetMacMethod ⌈

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 Transceiver Driver
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	

⌋(*SRS_Eth_00117*)

[SWS_EthIf_00474] ⌈The function EthIf_TransceiverGetMacMethod shall forward the call to function EthTrcv_GetMacMethod of the corresponding Ethernet Transceiver Driver (EthIfTransceiverIdx).⌋(*SRS_Eth_00117*)

[SWS_EthIf_00475] ⌈If development error detection is enabled: the function shall check that the service EthIf_Init was previously called. If the check fails, the function shall raise the development error ETHIF_E_UNINIT.⌋*()*

[SWS_EthIf_00476] ⌈If development error detection is enabled: the function shall check the parameter TrcvIdx for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_CTRL_IDX.⌋*()*

[SWS_EthIf_00477] ⌈If development error detection is enabled: the function shall check the parameter MacModePtr for being valid. If the check fails, the function shall raise the development error ETHIF_E_PARAM_POINTER.⌋*()*

8.3.85 EthIf_EthGetSpiStatus

[SWS_EthIf_91022]{DRAFT} **Definition of API function EthIf_EthGetSpiStatus** ⌈

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 controller Driver
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]{DRAFT} ⌈The function EthIf_EthGetSpiStatus shall forward the call to function Eth_GetSpiStatus of the corresponding Ethernet Driver (CtrlIdx).⌋*()*

[SWS_EthIf_00506]{DRAFT} ⌈If development error detection is enabled: the function shall check that the service EthIf_Init was previously called. If the check fails, the function shall raise the development error ETHIF_E_UNINIT.⌋*()*

[SWS_EthIf_00507]{DRAFT} ⌈If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_CTRL_IDX.⌋*()*

[SWS_EthIf_00508]{DRAFT} ⌈If development error detection is enabled: the function shall check the parameter SpiStatusPtr for being valid. If the check fails, the function shall raise the development error ETHIF_E_PARAM_POINTER.⌋*()*

8.3.86 EthIf_GetBufCV2xPC5RxParams

[SWS_EthIf_91201] Definition of API function EthIf_GetBufCV2xPC5RxParams ⌈

Service Name	EthIf_GetBufCV2xPC5RxParams	
Syntax	`Std_ReturnType EthIf_GetBufCV2xPC5RxParams ( uint8 CtrlId, const CV2x_BufCV2xPC5RxParamIdType* RxParamIds, uint16* ParamValues, uint8 NumParams )`	
Service ID [hex]	0x60	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	CtrlId	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 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]{DRAFT} ⌈The function `EthIf_GetBufCV2xPC5RxParams` shall forward the call to function `CV2x_GetBufCV2xPC5RxParams` of the respective Cellular V2X Driver.⌋*()*

[SWS_EthIf_00522]{DRAFT} ⌈The function `EthIf_GetBufCV2xPC5RxParams` shall be pre compile time configurable On/Off by the configuration parameter: `EthIfEnableCV2xApi`.⌋*()*

[SWS_EthIf_00523]{DRAFT} ⌈If development error detection is enabled: the function shall check that the service `EthIf_Init` was previously called. If the check fails, the function shall raise the development error `ETHIF_E_UNINIT`.⌋*()*

[SWS_EthIf_00524]{DRAFT} ⌈If development error detection is enabled: the function shall check the parameter `CtrlIdx` for being valid. If the check fails, the function shall raise the development error `ETHIF_E_INV_CTRL_IDX`.⌋*()*

[SWS_EthIf_00525]{DRAFT} ⌈If development error detection is enabled: the function shall check the parameter `RxParamIds` for being valid. If the check fails, the function shall raise the development error `ETHIF_E_PARAM_POINTER`.⌋*()*

[SWS_EthIf_00526]{DRAFT} ⌈If development error detection is enabled: the function shall check the parameter `ParamValues` for being valid. If the check fails, the function shall raise the development error `ETHIF_E_PARAM_POINTER`.⌋*()* Note: The function requires previous transmission (`EthIf_RxIndication`).

8.3.87 EthIf_GetBufCV2xPC5TxParams

[SWS_EthIf_91202] Definition of API function EthIf_GetBufCV2xPC5TxParams ⌈

Service Name	EthIf_GetBufCV2xPC5TxParams	
Syntax	`Std_ReturnType EthIf_GetBufCV2xPC5TxParams ( uint8 CtrlId, const CV2x_BufCV2xPC5TxParamIdType* TxParamIds, uint16* ParamValues, uint8 NumParams )`	
Service ID [hex]	0x61	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	CtrlId	Index of the Ethernet controller within the context of the Ethernet Interface
	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]{DRAFT} ⌈The function `EthIf_GetBufCV2xPC5TxParams` shall forward the call to function `CV2x_GetBufCV2xPC5TxParams` of the respective Cellular V2X Driver.⌋()

[SWS_EthIf_00532]{DRAFT} ⌈The function `EthIf_GetBufCV2xPC5TxParams` shall be pre compile time configurable On/Off by the configuration parameter: `EthIfEnableCV2xApi`.⌋()

[SWS_EthIf_00533]{DRAFT} ⌈If development error detection is enabled: the function shall check that the service `EthIf_Init` was previously called. If the check fails, the function shall raise the development error `ETHIF_E_UNINIT`.⌋()

[SWS_EthIf_00534]{DRAFT} ⌈If development error detection is enabled: the function shall check the parameter `CtrlIdx` for being valid. If the check fails, the function shall raise the development error `ETHIF_E_INV_CTRL_IDX`.⌋()

[SWS_EthIf_00535]{DRAFT} ⌈If development error detection is enabled: the function shall check the parameter `TxParamIds` for being valid. If the check fails, the function shall raise the development error `ETHIF_E_PARAM_POINTER`.⌋()

[SWS_EthIf_00536]{DRAFT} ⌈If development error detection is enabled: the function shall check the parameter `ParamValues` for being valid. If the check fails, the function shall raise the development error `ETHIF_E_PARAM_POINTER`.⌋() Note: The function requires previous transmission (`EthIf_Transmit`).

8.3.88 EthIf_SetBufCV2xPC5TxParams

[SWS_EthIf_91203] Definition of API function EthIf_SetBufCV2xPC5TxParams ⌈

Service Name	EthIf_SetBufCV2xPC5TxParams	
Syntax	`Std_ReturnType EthIf_SetBufCV2xPC5TxParams ( uint8 CtrlId, uint8 BufIdx, const CV2x_BufCV2xPC5TxParamIdType* TxParamIds, const uint16* ParamValues, uint8 NumParams )`	
Service ID [hex]	0x62	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	CtrlId	Index of the Ethernet controller within the context of the Ethernet Interface
	BufIdx	Index of the buffer resource
	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]{DRAFT} ⌈The function `EthIf_SetBufCV2xPC5TxParams` shall forward the call to function `CV2x_SetBufCV2xPC5TxParams` of the respective Cellular V2X Driver.⌋()

[SWS_EthIf_00542]{DRAFT} ⌈The function `EthIf_SetBufCV2xPC5TxParams` shall be pre compile time configurable On/Off by the configuration parameter: `EthIfEnableCV2xApi`.⌋()

[SWS_EthIf_00543]{DRAFT} ⌈If development error detection is enabled: the function shall check that the service `EthIf_Init` was previously called. If the check fails, the function shall raise the development error `ETHIF_E_UNINIT`.⌋()

[SWS_EthIf_00544]{DRAFT} ⌈If development error detection is enabled: the function shall check the parameter `CtrlIdx` for being valid. If the check fails, the function shall raise the development error `ETHIF_E_INV_CTRL_IDX`.⌋()

[SWS_EthIf_00545]{DRAFT} ⌈If development error detection is enabled: the function shall check the parameter `BufIdx` for being valid. If the check fails, the function shall raise the development error `ETHIF_E_PARAM_POINTER`.⌋()

[SWS_EthIf_00546]{DRAFT} ⌈If development error detection is enabled: the function shall check the parameter `TxParamIds` for being valid. If the check fails, the function shall raise the development error `ETHIF_E_PARAM_POINTER`.⌋()

[SWS_EthIf_00547]{DRAFT} ⌈If development error detection is enabled: the function shall check the parameter ParamValues for being valid. If the check fails, the function shall raise the development error ETHIF_E_PARAM_POINTER.⌋*()*

Note: The function requires previous transmission (EthIf_ProvideTxBuffer).

8.3.89 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]	0x63	
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]{DRAFT} ⌈The function EthIf_GetChanCV2xPC5TxParams shall forward the call to function Cv2x_GetChanTxParams of the respective Cellular V2X Driver.⌋*()*

[SWS_EthIf_00552]{DRAFT} ⌈The function EthIf_GetChanCV2xPC5TxParams shall be pre compile time configurable On/Off by the configuration parameter: EthIfEnableCV2xApi.⌋*()*

[SWS_EthIf_00553]{DRAFT} ⌈If development error detection is enabled: the function shall check that the service EthIf_Init was previously called. If the check fails, the function shall raise the development error ETHIF_E_UNINIT.⌋*()*

[SWS_EthIf_00554]{DRAFT} ⌈If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_CTRL_IDX.⌋*()*

[SWS_EthIf_00555]{DRAFT} ⌈If development error detection is enabled: the function shall check the parameter `ChannelId` for being valid. If the check fails, the function shall raise the development error `ETHIF_E_INV_PARAM`.⌋*()*

[SWS_EthIf_00556]{DRAFT} ⌈If development error detection is enabled: the function shall check the parameter `ParamIds` for being valid. If the check fails, the function shall raise the development error `ETHIF_E_PARAM_POINTER`.⌋*()*

[SWS_EthIf_00557]{DRAFT} ⌈If development error detection is enabled: the function shall check the parameter `ParamValues` for being valid. If the check fails, the function shall raise the development error `ETHIF_E_PARAM_POINTER`.⌋*()*

8.3.90 EthIf_SwitchMacSecUpdateSecY

[SWS_EthIf_91219]{DRAFT} **Definition of API function EthIf_SwitchMacSecUpdateSecY** ⌈

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 EthIf, PortIdx in context of EthSwt.
	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	

⌋*()*

8.3.91 EthIf_MacSecUpdateSecY

[SWS_EthIf_91215]{DRAFT} **Definition of API function EthIf_MacSecUpdateSecY**
⌈

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	

⌋()

8.3.92 EthIf_SwitchMacSecUpdateSecYNotification

[SWS_EthIf_91217]{DRAFT} **Definition of callback function EthIf_SwitchMacSec UpdateSecYNotification** ⌈

Service Name	EthIf_SwitchMacSecUpdateSecYNotification (DRAFT)	
Syntax	`void EthIf_SwitchMacSecUpdateSecYNotification ( const EthSwt_MgmtInfoType* MgmtInfoPtr )`	
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 EthIf, PortIdx in context of EthSwt.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	

▽

△

Description	Callback to notify that EhtIf_SwitchMacSecUpdateSecY finished. **Tags:** atp.Status=DRAFT
Available via	EthIf.h

⌋()

8.3.93 EthIf_MacSecUpdateSecYNotification

[SWS_EthIf_91218]{DRAFT} **Definition of callback function EthIf_MacSecUpdate SecYNotification** ⌈

Service Name	EthIf_MacSecUpdateSecYNotification (DRAFT)	

Syntax	`void EthIf_MacSecUpdateSecYNotification (`
	`  uint8 CtrlIdx`
	`)`	

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
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Callback to notify that EhtIf_SwitchMacSecUpdateSecY finished. **Tags:** atp.Status=DRAFT	
Available via	EthIf.h	

⌋()

8.3.94 EthIf_SwitchMacSecInitRxSc

[SWS_EthIf_91220]{DRAFT} **Definition of API function EthIf_SwitchMacSecInitRx Sc** ⌈

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 EthIf, PortIdx in context of EthSwt.

▽

△

	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	

⌋()

8.3.95 EthIf_MacSecInitRxSc

[SWS_EthIf_91211]{DRAFT} **Definition of API function EthIf_MacSecInitRxSc** ⌈

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	

⌋()

8.3.96 EthIf_SwitchMacSecResetRxSc

[SWS_EthIf_91221]{DRAFT} **Definition of API function EthIf_SwitchMacSecReset RxSc** ⌈

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 EthIf, PortIdx in context of EthSwt.
	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	

⌋()

8.3.97 EthIf_MacSecResetRxSc

[SWS_EthIf_91213]{DRAFT} **Definition of API function EthIf_MacSecResetRxSc** ⌈

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

⌋()

8.3.98 EthIf_SwitchMacSecAddTxSa

[SWS_EthIf_91222]{DRAFT} **Definition of API function EthIf_SwitchMacSecAddTxSa** ⌈

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 EthIf, PortIdx in context of EthSwt.
	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	

⌋()

8.3.99 EthIf_MacSecAddTxSa

[SWS_EthIf_91206]{DRAFT} **Definition of API function EthIf_MacSecAddTxSa** ⌈

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	

⌋()

8.3.100 EthIf_SwitchMacSecAddTxSaNotification

[SWS_EthIf_91223]{DRAFT} **Definition of callback function EthIf_SwitchMacSec AddTxSaNotification** ⌈

Service Name	EthIf_SwitchMacSecAddTxSaNotification (DRAFT)

Syntax	`void EthIf_SwitchMacSecAddTxSaNotification (`
	`  const EthSwt_MgmtInfoType* MgmtInfoPtr`
	`)`

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 EthIf, PortIdx in context of EthSwt.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Callback to notify that EthIf_SwitchMacSecAddTxSa finished. **Tags:** atp.Status=DRAFT	
Available via	EthIf.h	

⌋()

8.3.101 EthIf_MacSecAddTxSaNotification

[SWS_EthIf_91224]{DRAFT} **Definition of callback function EthIf_MacSecAddTx SaNotification** ⌈

Service Name	EthIf_MacSecAddTxSaNotification (DRAFT)	

Syntax	`void EthIf_MacSecAddTxSaNotification (`
	`  uint8 CtrlIdx`
	`)`	

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
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Callback to notify that EthIf_MacSecAddTxSa finished. **Tags:** atp.Status=DRAFT	
Available via	EthIf.h	

⌋()

8.3.102 EthIf_SwitchMacSecUpdateTxSa

[SWS_EthIf_91225]{DRAFT} **Definition of API function EthIf_SwitchMacSecUpdateTxSa** ⌈

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 EthIf, PortIdx in context of EthSwt.
	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	

⌋()

8.3.103 EthIf_MacSecUpdateTxSa

[SWS_EthIf_91216]{DRAFT} **Definition of API function EthIf_MacSecUpdateTxSa** ⌈

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	

⌋()

8.3.104 EthIf_SwitchMacSecDeleteTxSa

[SWS_EthIf_91226]{DRAFT} **Definition of API function EthIf_SwitchMacSecDeleteTxSa** ⌈

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 EthIf, PortIdx in context of EthSwt.
	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	

⌋()

8.3.105 EthIf_MacSecDeleteTxSa

[SWS_EthIf_91208]{DRAFT} **Definition of API function EthIf_MacSecDeleteTxSa** ⌈

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	

⌋()

8.3.106 EthIf_SwitchMacSecAddRxSa

[SWS_EthIf_91227]{DRAFT} **Definition of API function EthIf_SwitchMacSecAdd RxSa** ⌈

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 EthIf, PortIdx in context of EthSwt.
	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	

⌋()

8.3.107 EthIf_MacSecAddRxSa

[SWS_EthIf_91205]{DRAFT} **Definition of API function EthIf_MacSecAddRxSa** ⌈

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	

⌋()

8.3.108 EthIf_SwitchMacSecAddRxSaNotification

[SWS_EthIf_91228]{DRAFT} **Definition of callback function EthIf_SwitchMacSecAddRxSaNotification** ⌈

Service Name	EthIf_SwitchMacSecAddRxSaNotification (DRAFT)	
Syntax	`void EthIf_SwitchMacSecAddRxSaNotification ( const EthSwt_MgmtInfoType* MgmtInfoPtr )`	
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 EthIf, PortIdx in context of EthSwt.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Callback to notify that EthIf_SwitchMacSecAddRxSa finished. **Tags:** atp.Status=DRAFT	
Available via	EthIf.h	

⌋()

8.3.109 EthIf_MacSecAddRxSaNotification

[SWS_EthIf_91229]{DRAFT} **Definition of callback function EthIf_MacSecAddRxSaNotification** ⌈

Service Name	EthIf_MacSecAddRxSaNotification (DRAFT)
Syntax	`void EthIf_MacSecAddRxSaNotification ( uint8 CtrlIdx )`
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
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Callback to notify that EthIf_MacSecAddRxSa finished. **Tags:** atp.Status=DRAFT	
Available via	EthIf.h	

⌋()

8.3.110 EthIf_SwitchMacSecUpdateRxSa

[SWS_EthIf_91230]{DRAFT} **Definition of API function EthIf_SwitchMacSecUpdateRxSa** ⌈

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 EthIf, PortIdx in context of EthSwt.
	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	

⌋()

8.3.111 EthIf_MacSecUpdateRxSa

[SWS_EthIf_91214]{DRAFT} **Definition of API function EthIf_MacSecUpdateRxSa**
⌈

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	

⌋()

8.3.112 EthIf_SwitchMacSecDeleteRxSa

[SWS_EthIf_91231]{DRAFT} **Definition of API function EthIf_SwitchMacSecDelete RxSa** ⌈

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 EthIf, PortIdx in context of EthSwt.

▽

△

	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	

⌋()

8.3.113 EthIf_MacSecDeleteRxSa

[SWS_EthIf_91207]{DRAFT} **Definition of API function EthIf_MacSecDeleteRxSa**
⌈

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	

⌋()

8.3.114 EthIf_SwitchMacSecGetTxSaNextPn

[SWS_EthIf_91232]{DRAFT} **Definition of API function EthIf_SwitchMacSecGetTxSaNextPn** ⌈

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 EthIf, PortIdx in context of EthSwt.
	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	

⌋()

8.3.115 EthIf_MacSecGetTxSaNextPn

[SWS_EthIf_91210]{DRAFT} **Definition of API function EthIf_MacSecGetTxSaNextPn** ⌈

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	

⌋()

8.3.116 EthIf_SwitchMacSecGetMacSecStats

[SWS_EthIf_91233]{DRAFT} **Definition of API function EthIf_SwitchMacSecGetMacSecStats** ⌈

Service Name	EthIf_SwitchMacSecGetMacSecStats (DRAFT)	
Syntax	`Std_ReturnType EthIf_SwitchMacSecGetMacSecStats ( const EthSwt_MgmtInfoType* MgmtInfoPtr )`	
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 EthIf, PortIdx in context of EthSwt.
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 provide MACsec statistics. The result is returned through EthIf_SwitchMacSecGetMacSecStatsNotification. **Tags:** atp.Status=DRAFT	
Available via	EthIf.h	

⌋()

8.3.117 EthIf_MacSecGetMacSecStats

[SWS_EthIf_91209]{DRAFT} **Definition of API function EthIf_MacSecGetMacSecStats** ⌈

Service Name	EthIf_MacSecGetMacSecStats (DRAFT)	

Syntax	`Std_ReturnType EthIf_MacSecGetMacSecStats (`
	`  uint8 CtrlIdx`
	`)`	

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)	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 provide MACsec statistics. The result is returned through EthIf_MacSecGetMacSecStatsNotification **Tags:** atp.Status=DRAFT	
Available via	EthIf.h	

⌋()

8.3.118 EthIf_SwitchMacSecGetMacSecStatsNotification

[SWS_EthIf_91234]{DRAFT} **Definition of callback function EthIf_SwitchMacSecGetMacSecStatsNotification** ⌈

Service Name	EthIf_SwitchMacSecGetMacSecStatsNotification (DRAFT)	

Syntax	`void EthIf_SwitchMacSecGetMacSecStatsNotification (`
	`  const EthSwt_MgmtInfoType* MgmtInfoPtr,`
	`  const Mka_Stats_SecYType* MacSecStatsPtr`
	`)`	

Service ID [hex]	0x7c	
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 EthIf, PortIdx in context of EthSwt.
	MacSecStatsPtr	Pointer to a structure including the MACsec statistics of an MKA participant
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Callback to notify that EthIf_SwitchMacSecGetMacSecStats finished and provide the requested statistics. **Tags:** atp.Status=DRAFT	
Available via	EthIf.h	

⌋()

8.3.119 EthIf_MacSecGetMacSecStatsNotification

[SWS_EthIf_91235]{DRAFT} **Definition of callback function EthIf_MacSecGetMacSecStatsNotification** ⌈

Service Name	EthIf_MacSecGetMacSecStatsNotification (DRAFT)	

Syntax	`void EthIf_MacSecGetMacSecStatsNotification (`
	`  uint8 CtrlIdx,`
	`  const Mka_Stats_SecYType* MacSecStatsPtr`
	`)`	

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
	MacSecStatsPtr	Pointer to a structure including the MACsec statistics of an MKA participant
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Callback to notify that EthIf_MacSecGetMacSecStats finished and provide the requested statistics. **Tags:** atp.Status=DRAFT	
Available via	EthIf.h	

⌋()

8.3.120 EthIf_SwitchMacSecOperational

[SWS_EthIf_91236]{DRAFT} **Definition of API function EthIf_SwitchMacSecOperational** ⌈

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	
Parameters (in)	MgmtInfoPtr	Pointer to the management information within the context of an Ethernet Switch Driver. SwitchIdx in context of EthIf, PortIdx in context of EthSwt.
	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

⌋()

8.3.121 EthIf_MacSecOperational

[SWS_EthIf_91212]{DRAFT} **Definition of API function EthIf_MacSecOperational** ⌈

Service Name	EthIf_MacSecOperational (DRAFT)	

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

Service ID [hex]	0x1c	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
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	

⌋()

8.3.122 EthIf_SwitchMacSecSetControlledPortEnabled

[SWS_EthIf_91237]{DRAFT} **Definition of API function EthIf_SwitchMacSecSet ControlledPortEnabled** ⌈

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 EthIf, PortIdx in context of EthSwt.
	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	

⌋()

8.3.123 EthIf_MacSecSetControlledPortEnabled

[SWS_EthIf_91238]{DRAFT} **Definition of API function EthIf_MacSecSetControlledPortEnabled** ⌈

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	

⌋()

8.3.124 EthIf_GetStreamHandleIdxStatistics

[SWS_EthIf_91027]{DRAFT} **Definition of API function EthIf_GetStreamHandleIdx Statistics** ⌈

Service Name	EthIf_GetStreamHandleIdxStatistics (draft)	

Syntax	`void EthIf_GetStreamHandleIdxStatistics (`
	`  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	This function is called by the Firewall module to get bucket counter statistics from the switch. **Tags:** atp.Status=draft	
Available via	EthIf.h	

⌋(*FO_RS_Fw_00011*)

8.3.125 EthIf_SetStreamHandleIdxConfiguration

[SWS_EthIf_91025]{DRAFT} **Definition of API function EthIf_SetStreamHandleIdx Configuration** ⌈

Service Name	EthIf_SetStreamHandleIdxConfiguration (draft)	

Syntax	`void EthIf_SetStreamHandleIdxConfiguration (`
	`  uint8 SwitchIdx,`
	`  uint8 StreamHandleIdxPtr,`
	`  boolean StreamHandleIdxActivityStatus`
	`)`	

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
	StreamHandleIdxActivity Status	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 StreamHandleIdx in the switch. **Tags:** atp.Status=draft	
Available via	EthIf.h	

⌋(*FO_RS_Fw_00011*)

8.4 Callback notifications

This is a list of functions provided for other modules.

8.4.1 EthIf_RxIndication

[SWS_EthIf_00085] Definition of API function EthIf_RxIndication ⌈

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	

⌋*(SRS_Eth_00169)*

[SWS_EthIf_00086] ⌈If development error detection is enabled: the function shall check that the service `EthIf_Init` was previously called. If the check fails, the function shall raise the development error `ETHIF_E_UNINIT`.⌋*()*

[SWS_EthIf_00087] ⌈If development error detection is enabled: the function shall check the parameter `CtrlIdx` for being valid. If the check fails, the function shall raise the development error `ETHIF_E_INV_CTRL_IDX`.⌋*()*

[SWS_EthIf_00088] ⌈If development error detection is enabled: the function shall check the parameter DataPtr for being valid. If the check fails, the function shall raise the development error ETHIF_E_PARAM_POINTER.⌋()

[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⌋()

8.4.2 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.⌋()

[SWS_EthIf_00092] ⌈If development error detection is enabled: the function shall check that the service EthIf_Init was previously called. If the check fails, the function shall raise the development error ETHIF_E_UNINIT.⌋()

[SWS_EthIf_00093] ⌈If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_CTRL_IDX.⌋()

[SWS_EthIf_00094] ⌈If development error detection is enabled: the function shall check the parameter BufIdx for being valid. If the check fails, the function shall raise the development error ETHIF_E_INV_PARAM.⌋()

8.4.3 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`.⌋()

8.4.4 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	

⌋()

8.4.5 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	

⌋()

8.4.6 EthIf_SleepIndication

[SWS_EthIf_91006]{DRAFT} **Definition of API function EthIf_SleepIndication** ⌈

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	

⌋(*SRS_Eth_00156*)

[SWS_EthIf_00497]{DRAFT} ⌈The function shall call `EthSM_SleepIndication` with the corresponding EthIfCtrl.⌋*(SRS_Eth_00156)*

8.4.7 EthIf_StreamHandleIdxConfiguration

[SWS_EthIf_91024]{DRAFT} **Definition of callback function EthIf_StreamHandleIdxConfiguration** ⌈

Service Name	EthIf_StreamHandleIdxConfiguration (draft)	
Syntax	`void EthIf_StreamHandleIdxConfiguration ( uint8 SwitchIdx, uint8 StreamHandleIdxPtr, boolean StreamHandleIdxActivityStatus )`	
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
	StreamHandleIdxActivityStatus	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 EthSwtDrv once it has successfully set the StreamHandleIdx activity status in the switch. **Tags:** atp.Status=draft	
Available via	EthIf_Cbk.h	

⌋*(FO_RS_Fw_00011)*

8.4.8 EthIf_StreamHandleIdxStatistics

[SWS_EthIf_91023]{DRAFT} **Definition of callback function EthIf_StreamHandleIdxStatistics** ⌈

Service Name	EthIf_StreamHandleIdxStatistics (draft)	
Syntax	`void EthIf_StreamHandleIdxStatistics ( uint8 SwitchIdx, uint8 NumberOfBuckets, const uint8* StreamHandleIdxStatisticsPtr )`	
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
	StreamHandleIdx StatisticsPtr	Pointer to the bucket counter values
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	The function is called by the EthSwtDrv once it has successfully retrieved the bucket counter values from the switch. **Tags:** atp.Status=draft	
Available via	EthIf_Cbk.h	

⌋(*FO_RS_Fw_00011*)

8.5 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.

8.5.1 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`.⌋()

8.5.2 EthIf_MainFunctionRx_<PriorityProcessing ShortName>

[SWS_EthIf_91051]{OBSOLETE} **Definition of scheduled function EthIf_Main FunctionRx_<PriorityProcessing ShortName>** ⌈

Service Name	EthIf_MainFunctionRx_<PriorityProcessing ShortName> (obsolete)

Syntax	`void EthIf_MainFunctionRx_<PriorityProcessing ShortName> (`
	`  void`
	`)`

Service ID [hex]	0x42
Description	The function checks for new received frames at the related Ethernet controller or CanXL controller and reception queue by calling <EthDrv>_Receive() with the respective FifoIdx. Eth If_MainFunctionRx shall receive frames from all FIFOs that are not assigned for processing via EthIfPhysCtrlRxMainFunctionPriorityProcessing. **Tags:** atp.Status=obsolete
Available via	EthIf_SchM.h

⌋*()*

8.5.3 EthIf_MainFunctionRx_<IngressQueueProcessing ShortName>

[SWS_EthIf_91139]{DRAFT} **Definition of scheduled function EthIf_MainFunction Rx_<IngressQueueProcessing ShortName>** ⌈

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

⌋*(SRS_Eth_00170)*

8.5.4 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`.⌋()

8.5.5 EthIf_MainFunctionState

[SWS_EthIf_91104] Definition of API function EthIf_MainFunctionState ⌈

Service Name	EthIf_MainFunctionState

Syntax	`void EthIf_MainFunctionState (`
	`  void`
	`)`

Service ID [hex]	0x05
Sync/Async	Asynchronous
Reentrancy	Non Reentrant
Parameters (in)	None
Parameters (inout)	None
Parameters (out)	None
Return value	None
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]{DRAFT} ⌈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 EthIfQualifiedUnexpectedLinkDownTime, 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.⌋*(SRS_Eth_00156)*

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).

8.6 Expected interfaces

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

8.6.1 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 in module EthIf ⌈

API Function	Header File	Description
There are no mandatory interfaces.		

⌋*()*

8.6.2 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 in 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

▽

△

API Function	*Header File*	*Description*
CanXL_GetPhysAddr	CanXL.h	Obtains the physical source address used by the indexed controller
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_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
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_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

▽

△

API Function	*Header File*	*Description*
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_WriteMii	Eth.h	Configures a transceiver register or triggers a function offered by the receiver
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.
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_ExtractStreamHandleIdx (draft)	EthSwt.h	Extracts the StreamHandleIdx from the switch vendor specific part of the network packet header **Tags:** atp.Status=draft
EthSwt_GetStreamHandleIdxStatistics (draft)	EthSwt.h	Requests the bucket counter values from the switch **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.

▽

△

API Function	*Header File*	*Description*
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_SetStreamHandleIdx Configuration (draft)	EthSwt.h	This function is called by the EthIf module to control the activity status of a StreamHandleIdx in the switch. **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
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_InspectPacket (draft)	Fw.h	This function inspects the network packet against the list of pre-defined firewall rules and returns the inspection result within the InspectionResultPtr. **Tags:** atp.Status=draft
Fw_StreamHandleIdxConfiguration (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
Fw_StreamHandleIdxStatistics (draft)	Fw_Cbk.h	The function is called by the EthIf once it has successfully retrieved the bucket counter values from the switch. **Tags:** atp.Status=draft
IdsM_SetSecurityEvent	IdsM.h	This API is the application interface to report security events to the IdsM.
IdsM_SetSecurityEventWithContext Data	IdsM.h	This API is the application interface to report security events with context data to the IdsM.
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.
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.

▽

△

API Function	Header File	Description
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, ...).

⌋()

8.6.3 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.

[SWS_EthIf_00104] Definition of configurable interface <User>_RxIndication ⌈

Service Name	<User>_RxIndication	

Syntax	`void <User>_RxIndication (`
	`  uint8 CtrlIdx,`
	`  Eth_FrameType FrameType,`
	`  boolean IsBroadcast,`
	`  const uint8* PhysAddrPtr,`
	`  const uint8* DataPtr,`
	`  uint16 LenByte`
	`)`	

Sync/Async	Synchronous	
Reentrancy	Dont care	
Parameters (in)	CtrlIdx	Index of the 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 the received Ethernet frame (i.e. Ethernet header is not provided).
	LenByte	Length of received data.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Indicates the reception of an Ethernet frame	
Available via	configurable	

⌋()

[SWS_EthIf_00105] ⌈The callback function shall be configurable by the configuration parameter: `EthIfRxIndicationFunction`.⌋()

[SWS_EthIf_00106] Definition of configurable interface <UL>_TxConfirmation ⌈

Service Name	<UL>_TxConfirmation	
Syntax	`void <UL>_TxConfirmation ( uint8 CtrlIdx, Eth_BufIdxType BufIdx, Std_ReturnType Result )`	
Sync/Async	Synchronous	
Reentrancy	Dont care	
Parameters (in)	CtrlIdx	Index of the Ethernet controller within the context of the Ethernet Interface
	BufIdx	Index of the buffer resource
	Result	–
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Confirms the transmission of an Ethernet frame	
Available via	configurable	

⌋()

[SWS_EthIf_00107] ⌈The callback function shall be configurable by the configuration parameter: `EthIfTxConfirmationFunction`.⌋()

[SWS_EthIf_00108] Definition of configurable interface <User>_TrcvLinkState Chg ⌈

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_00229] ⌈EthIfControllers not referring to an Ethernet Transceiver, i.e. no valid `EthIfEthTrcvRef` or `EthIfCanXLTrcvRef` is configured, shall act as if the transceiver was present and the transceiver status was `ETHTRCV_LINK_STATE_ACTIVE`.⌋()

[SWS_EthIf_00230] ⌈Upon change of link state `<User>_TrcvLinkStateChg` shall be invoked for every affected EthIfController.⌋*()*

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).

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

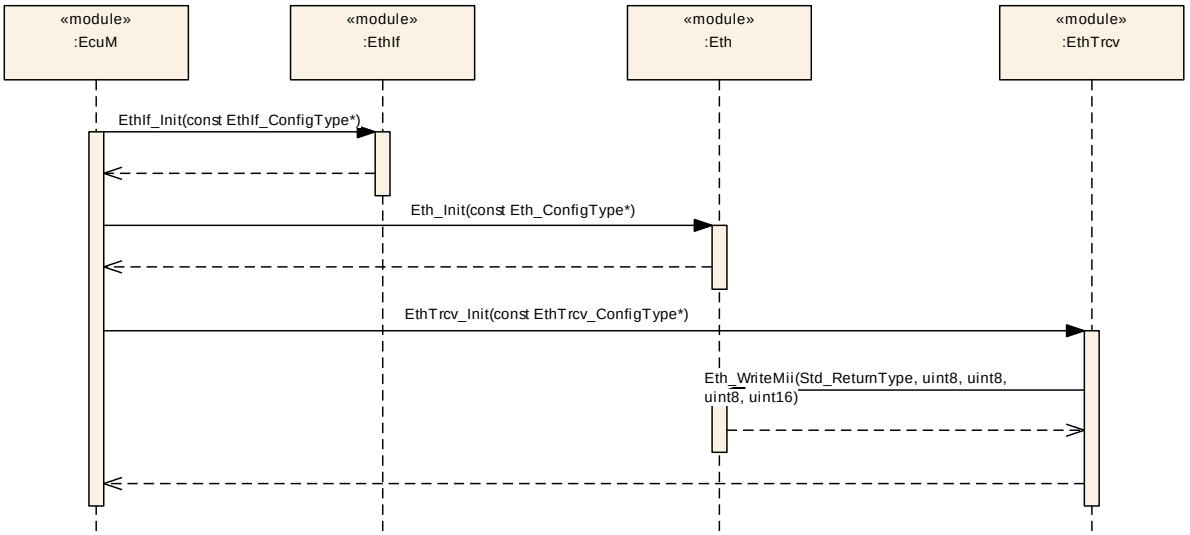

Figure 9.1: Initialization

9.2 Communication Initialization

Name: EthIf_CommunicationInitialization
Package: EthIf
Version: 1.0
Author: fix0ec2

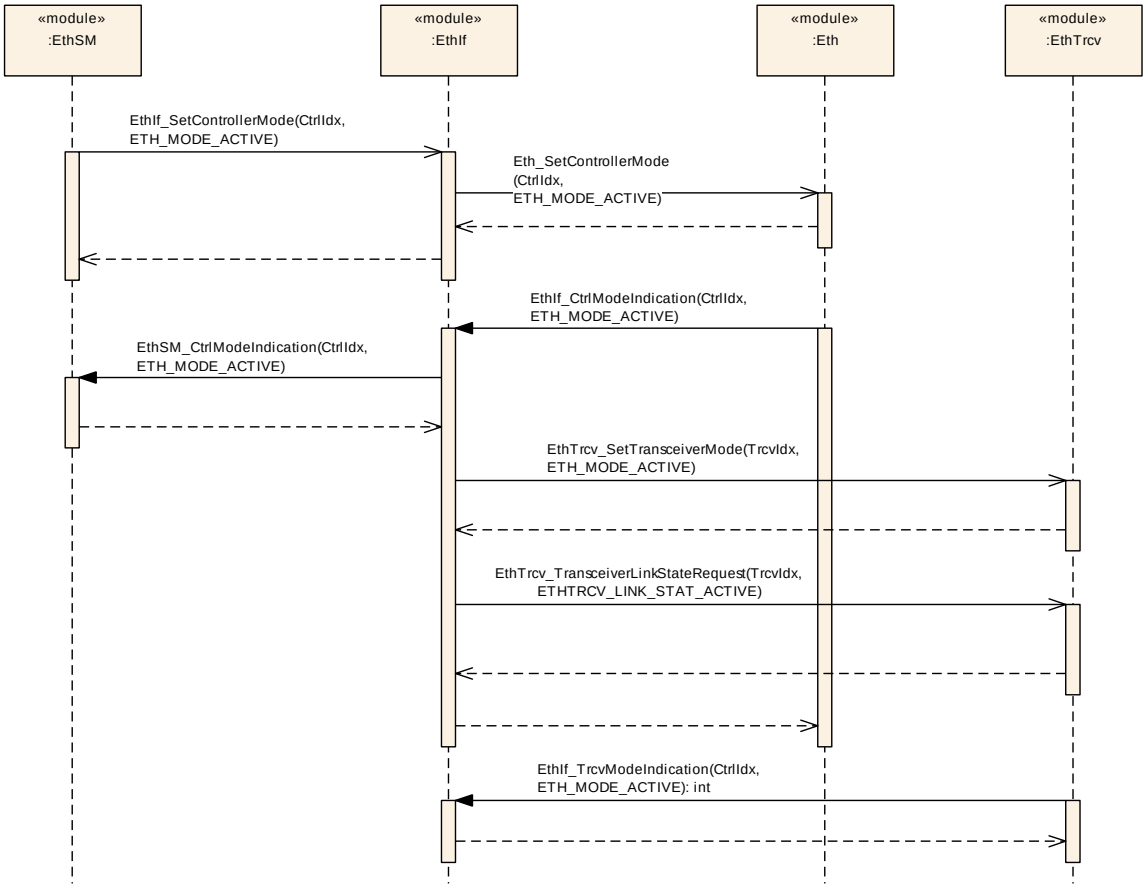

Figure 9.2: Communication Initialization

9.3 Switch Initialization

Name: EthIf_SwitchInitalization
Package: EthIf
Version: 1.0
Author: fix0ec2

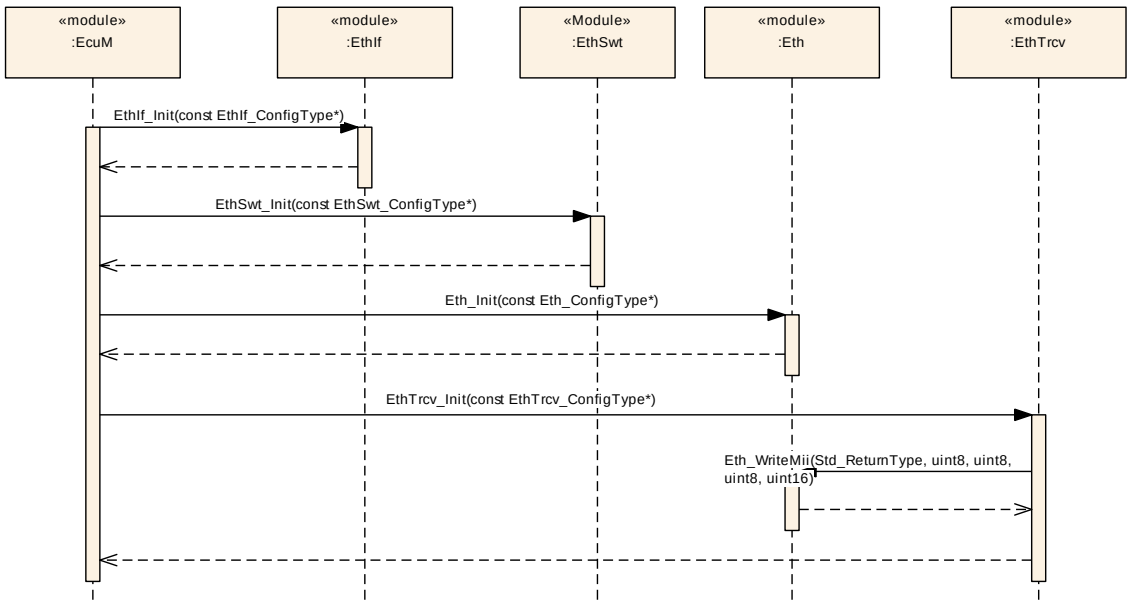

Figure 9.3: Switch Initialization

9.4 Data Transmission

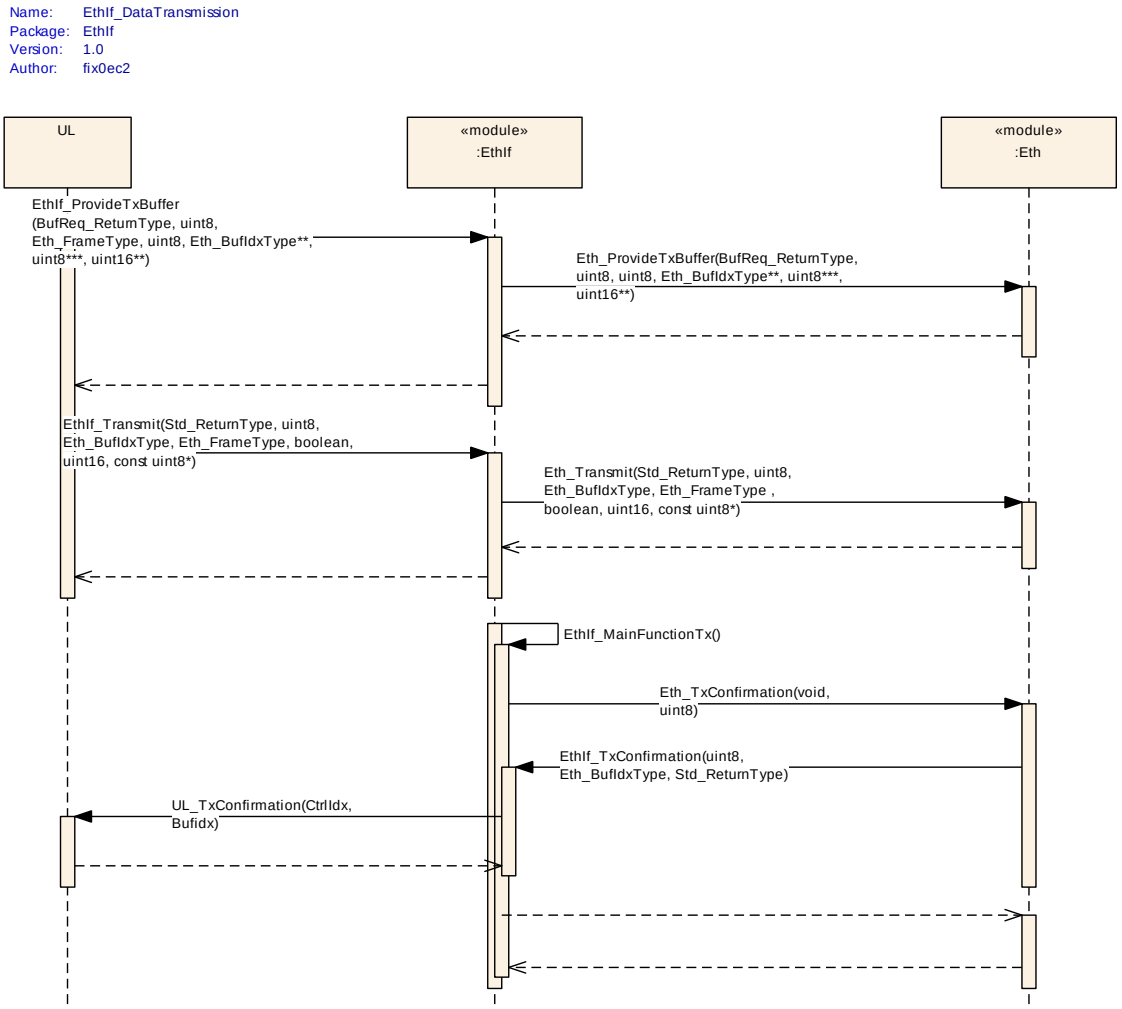

Figure 9.4: Frame Transmission in Polling Mode

[SWS_EthIf_00115] ⌈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`.⌋*()*

[SWS_EthIf_00125] ⌈`EthIf_TxConfirmation` shall forward the confirmation to the registered call-back functions <User>_TxConfirmation.⌋*()*

Name: EthIf_TransmissionInterrupt
Package: EthIf
Version: 1.0
Author: fix0ec2

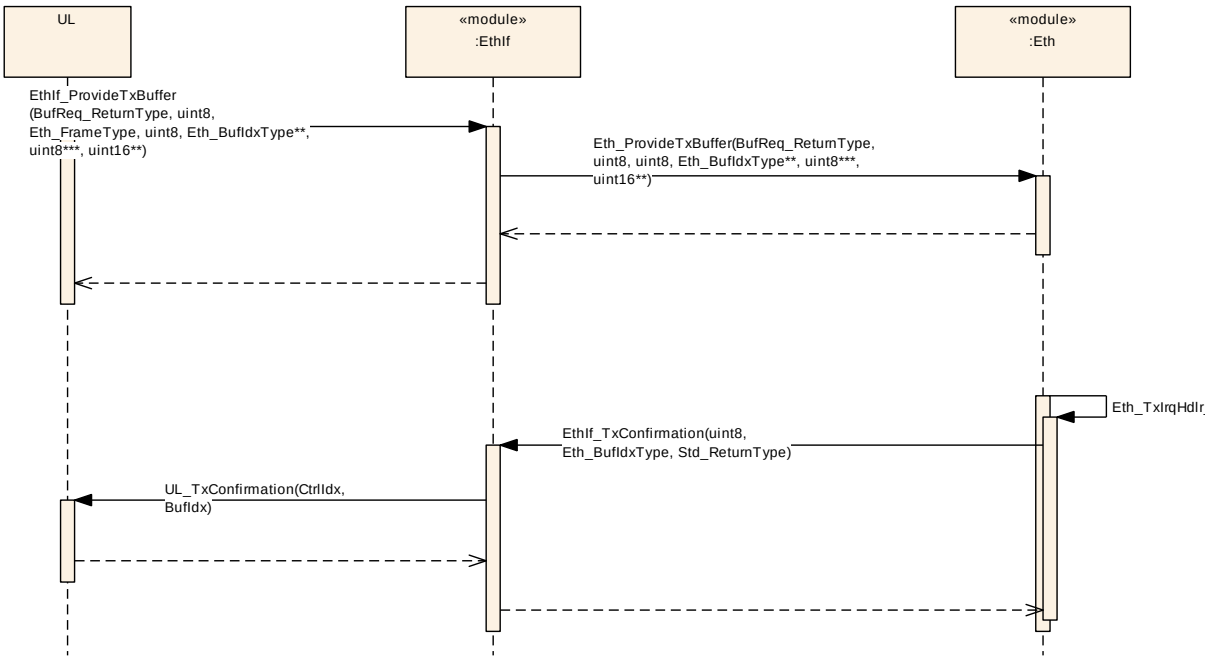

Figure 9.5: Frame Transmission in Interrupt Mode

9.5 Data Reception

Name: EthIf_DataReception
Package: EthIf
Version: 1.0
Author: fix0ec2

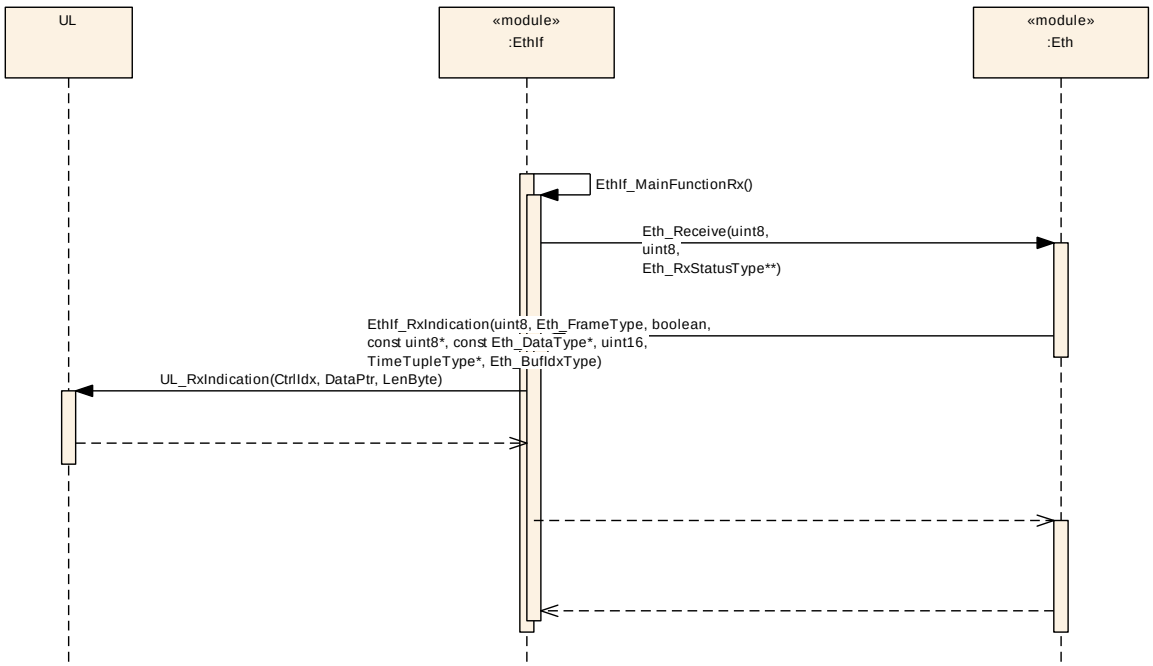

Figure 9.6: Frame Reception in Polling Mode

Name: EthIf_ReceptionInterrupt
Package: EthIf
Version: 1.0
Author: fix0ec2

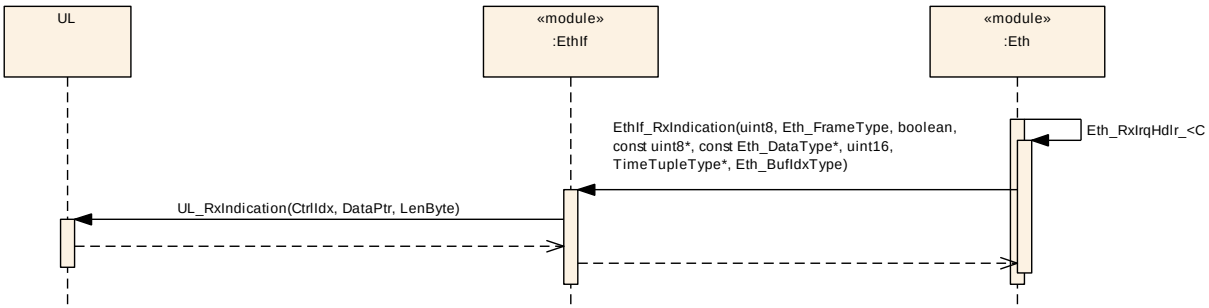

Figure 9.7: Frame Reception in Interrupt Mode

9.6 Link State Change

Name: EthIf_LinkStateChange
Package: EthIf
Version: 1.0
Author: fix0ec2

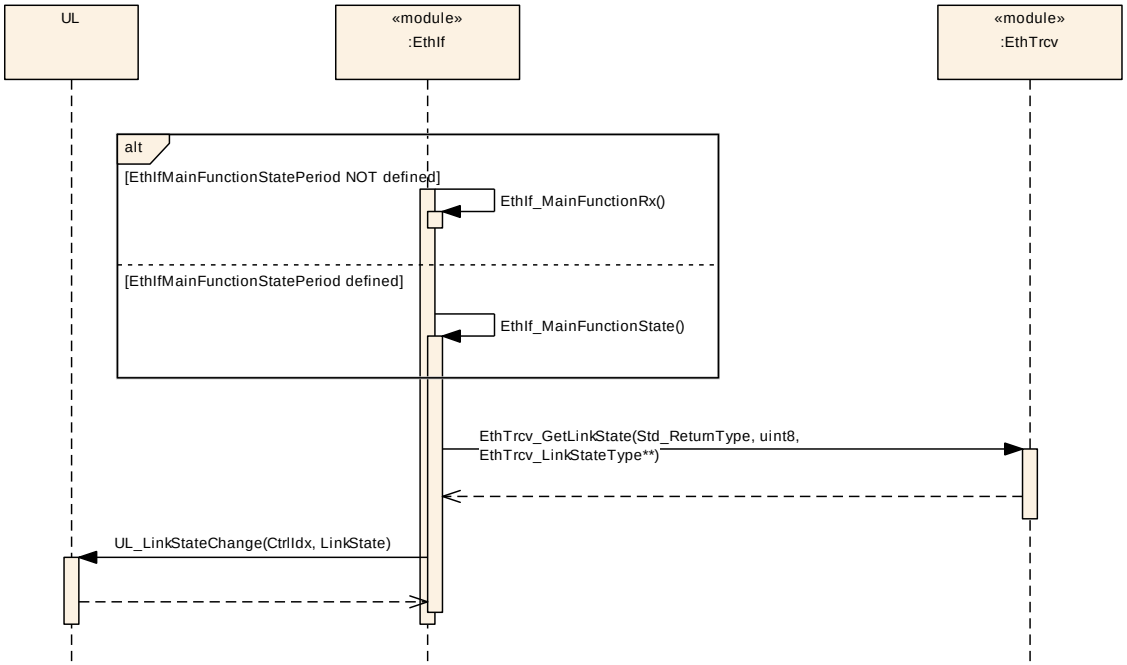

Figure 9.8: Link State Change

9.7 Link State Change without Port Groups

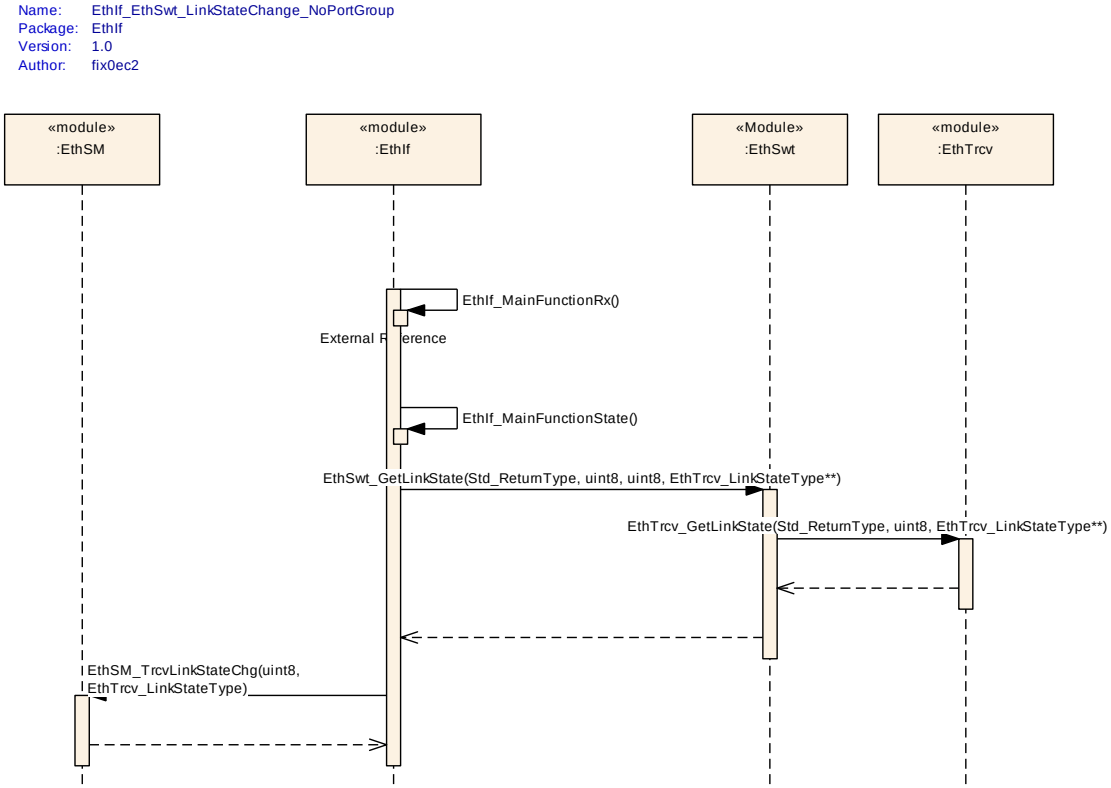

Figure 9.9: Link State Change without Port Groups

9.8 Link State Change with Port Groups

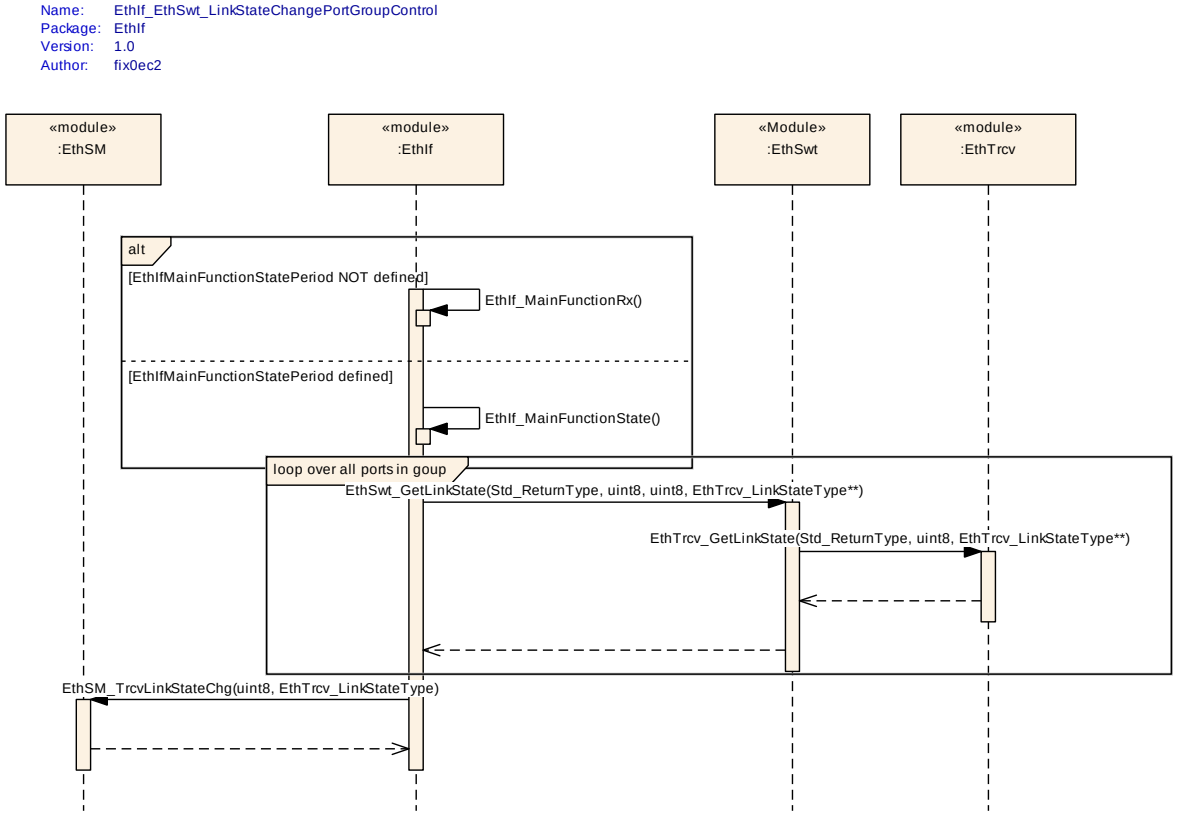

Figure 9.10: Link State Change with Port Groups

9.9 Link State Change with Port Groups and Partial Network Cluster

Name: EthIf_EthSwt_LinkStateChangePortGroupPNC
Package: EthIf
Version: 1.0
Author: fix0ec2

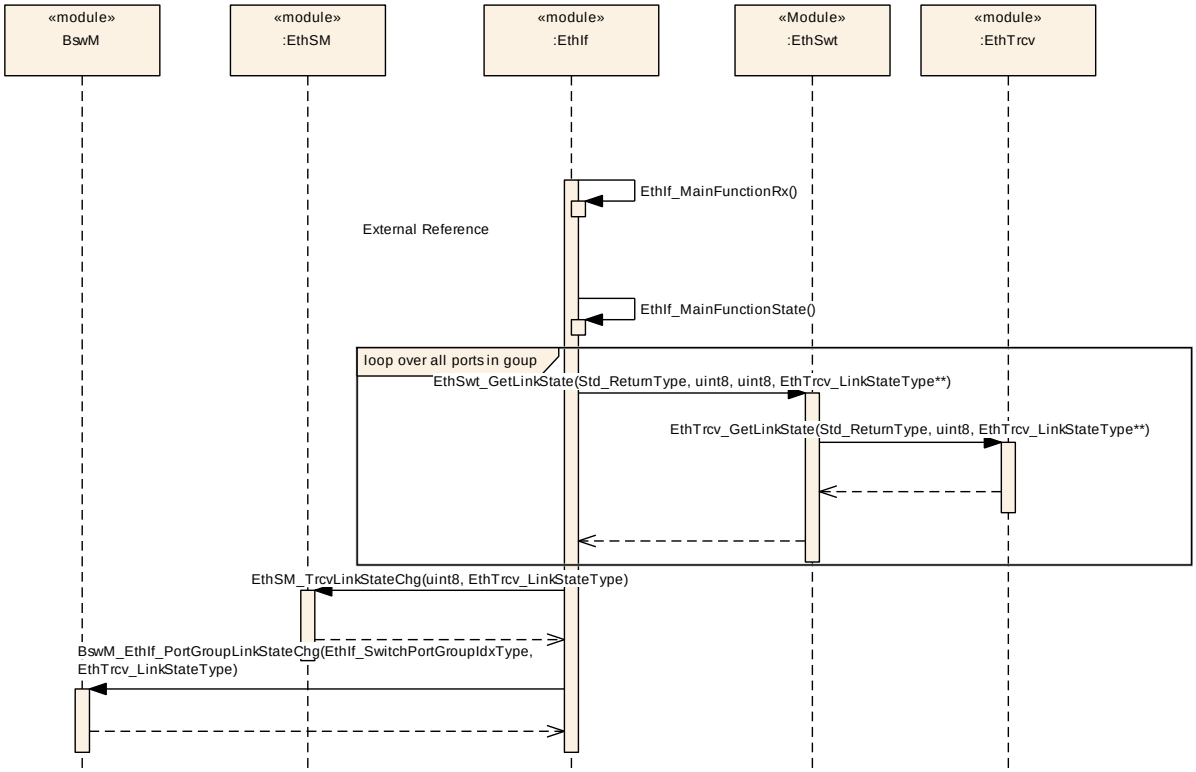

Figure 9.11: and Partial Network Cluster

9.10 Switch Management support

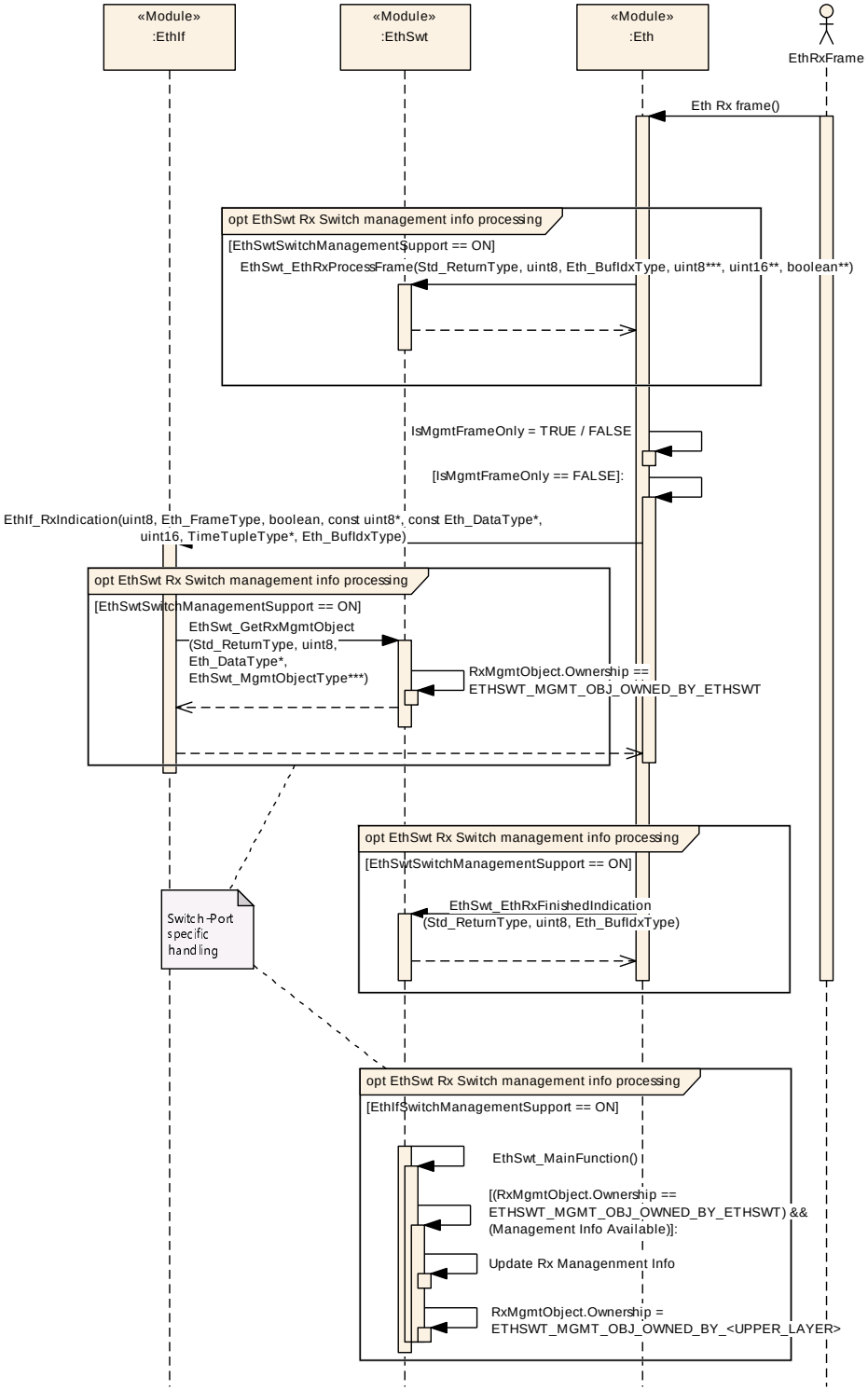

Figure 9.12: Switch Management support for transmission

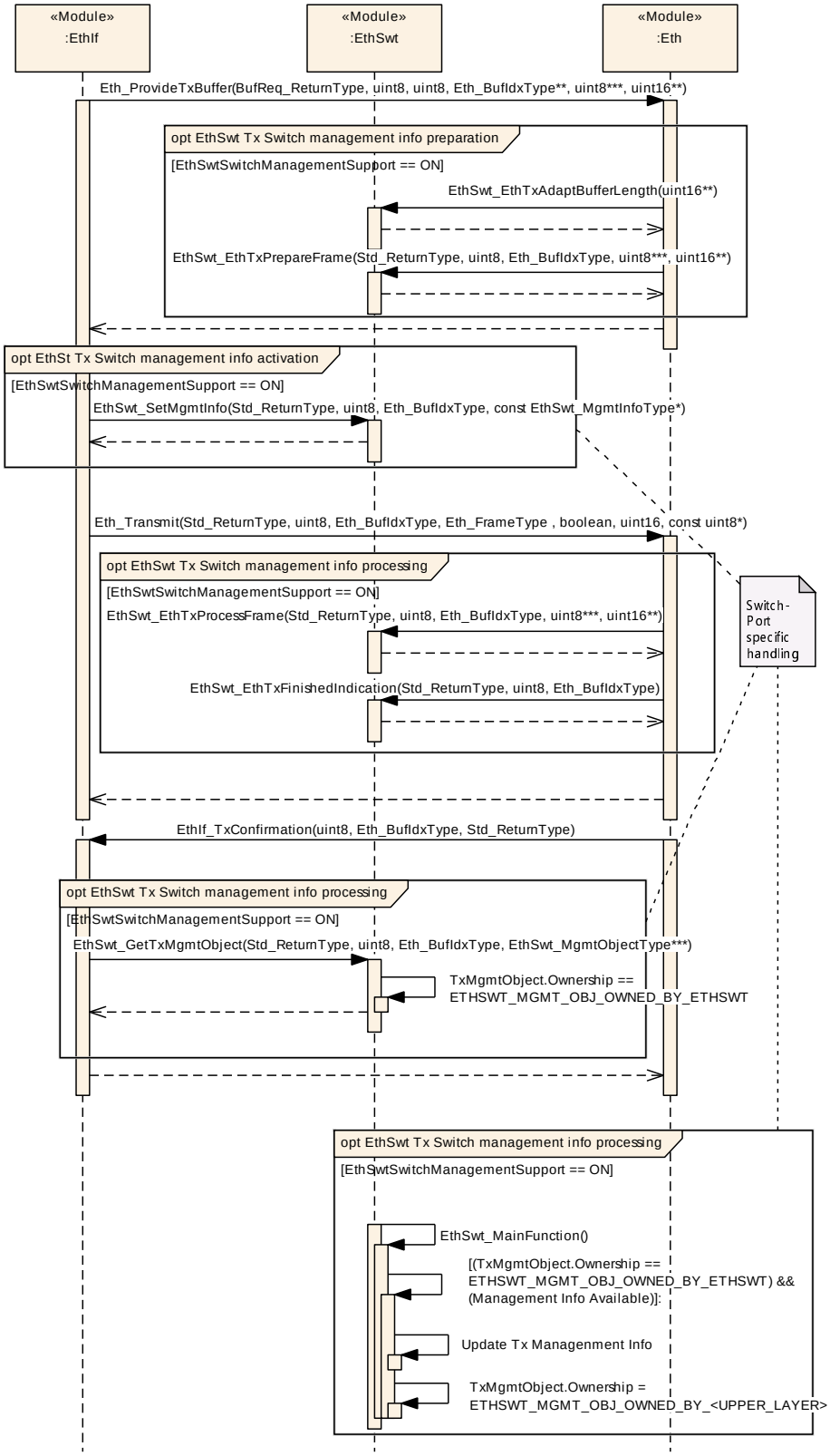

Figure 9.13: Switch Management support for reception

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 10.3 specifies published information of the module Ethernet Interface.

10.1 How to read this chapter

For details refer to the chapter 10.1 “Introduction to configuration specification” in SWS_BSWGeneral [6].

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.

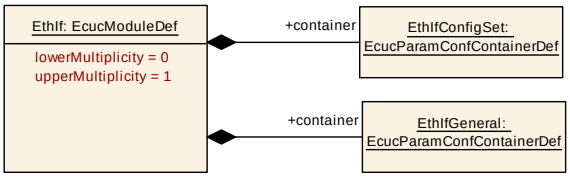

Figure 10.1: Ethernet Interface

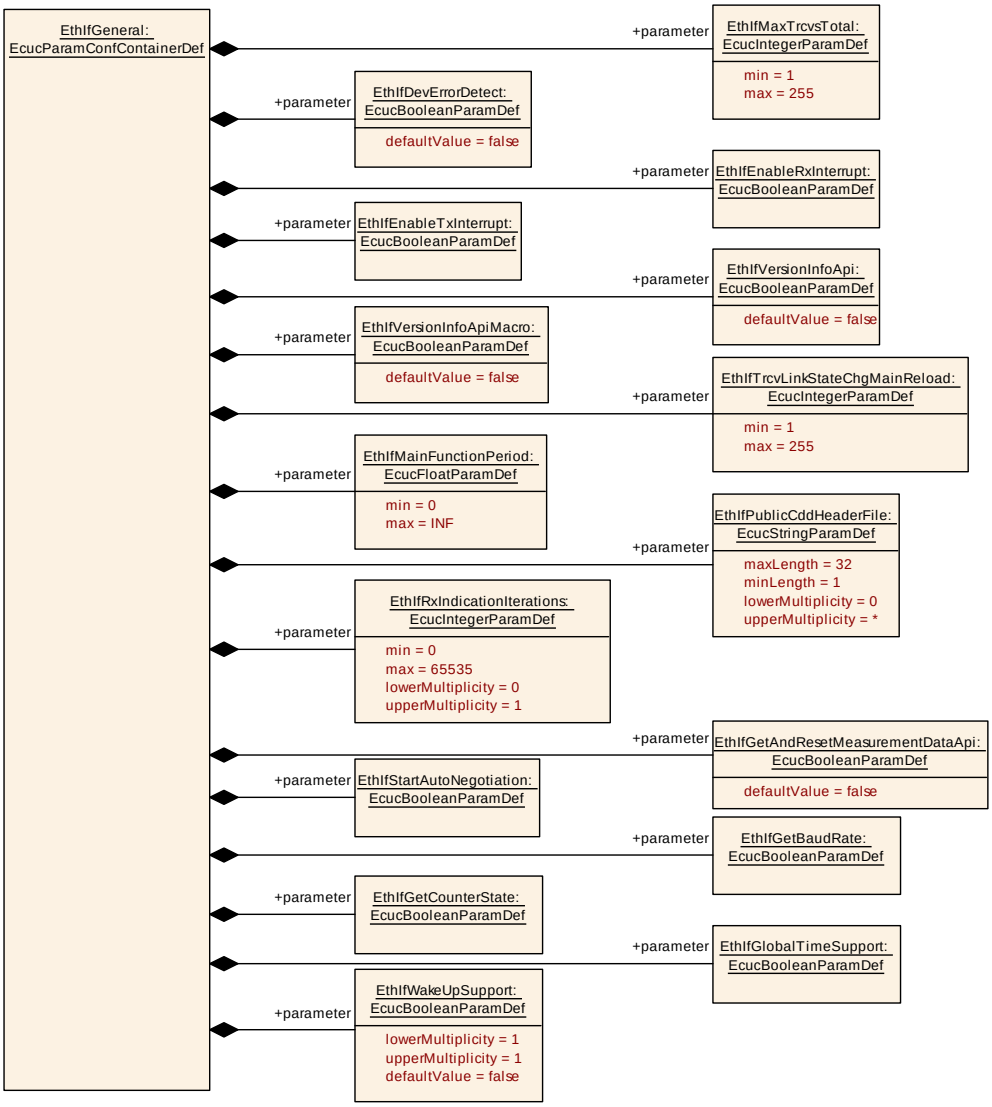

Figure 10.2: Ethernet Interface general configuration structure

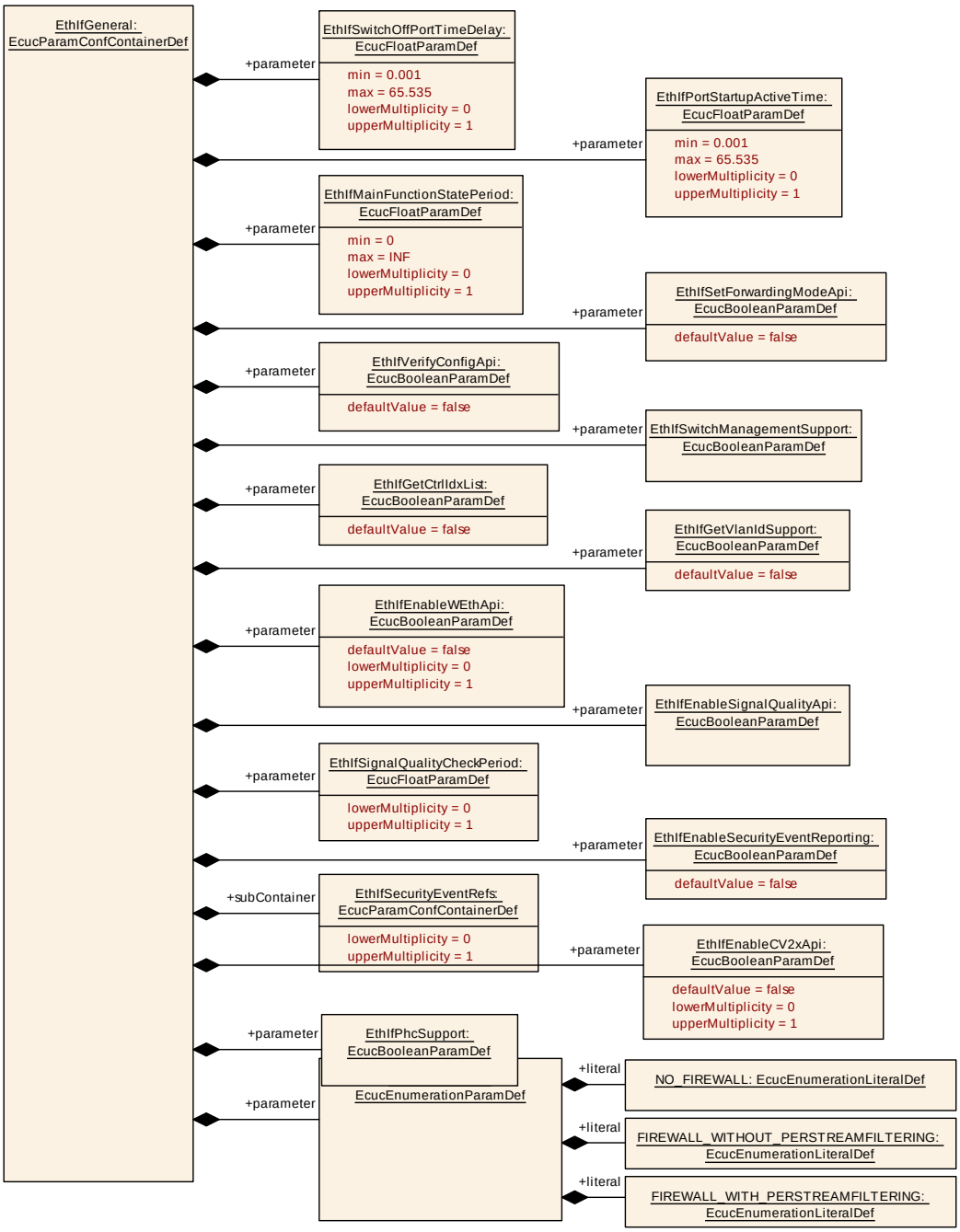

Figure 10.3: Ethernet Interface general configuration structure (continued)

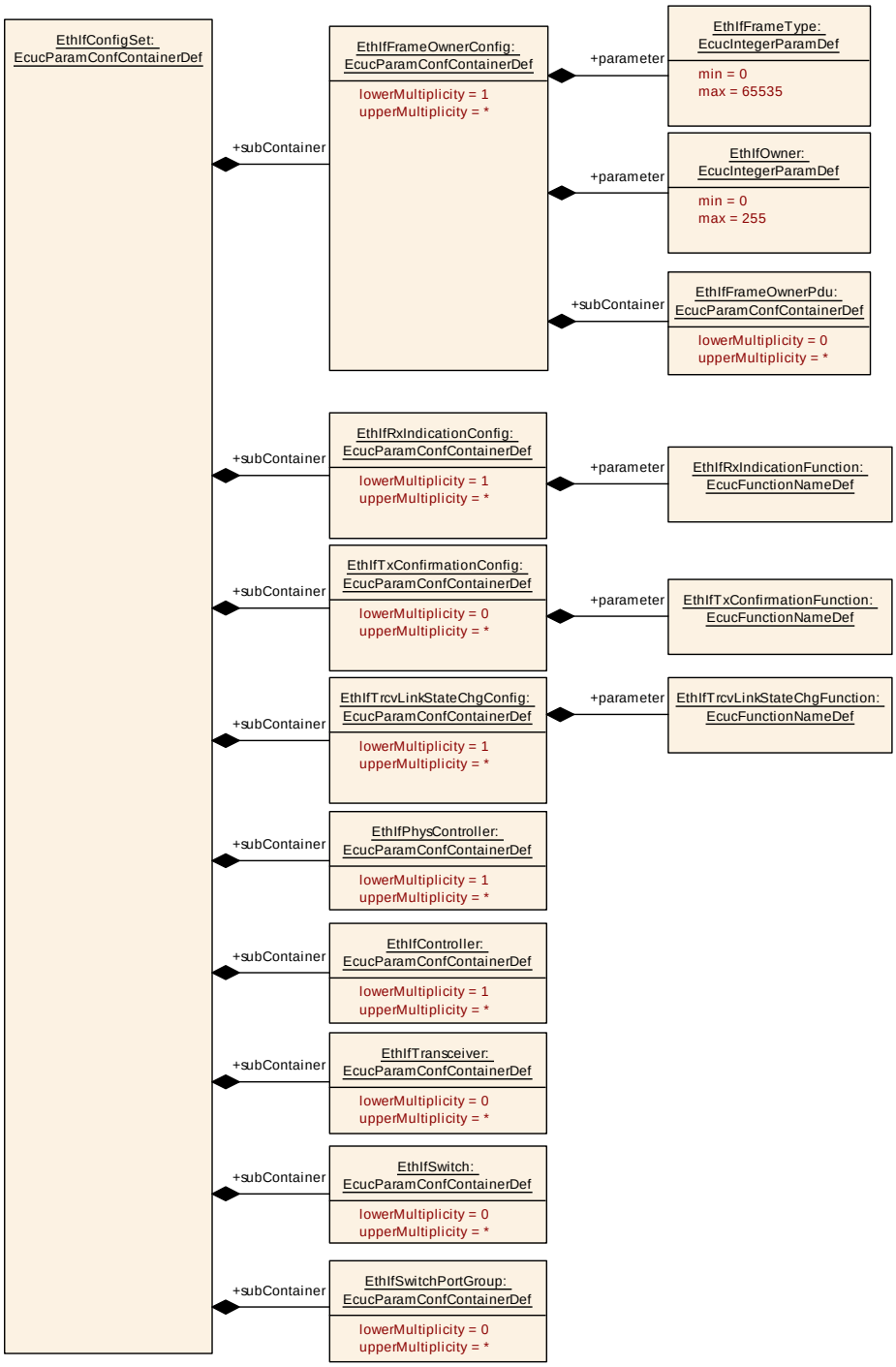

Figure 10.4: Ethernet Interface interface configuration structure

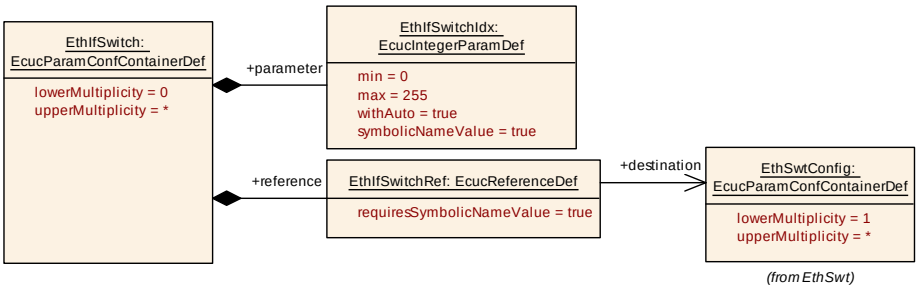

Figure 10.5: Ethernet Interface Switch configuration structure

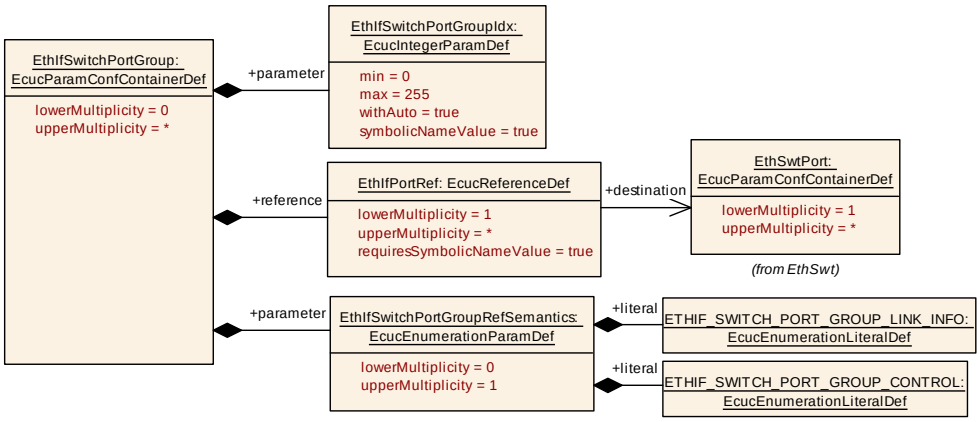

Figure 10.6: Ethernet Interface SwitchPortGroup configuration structure

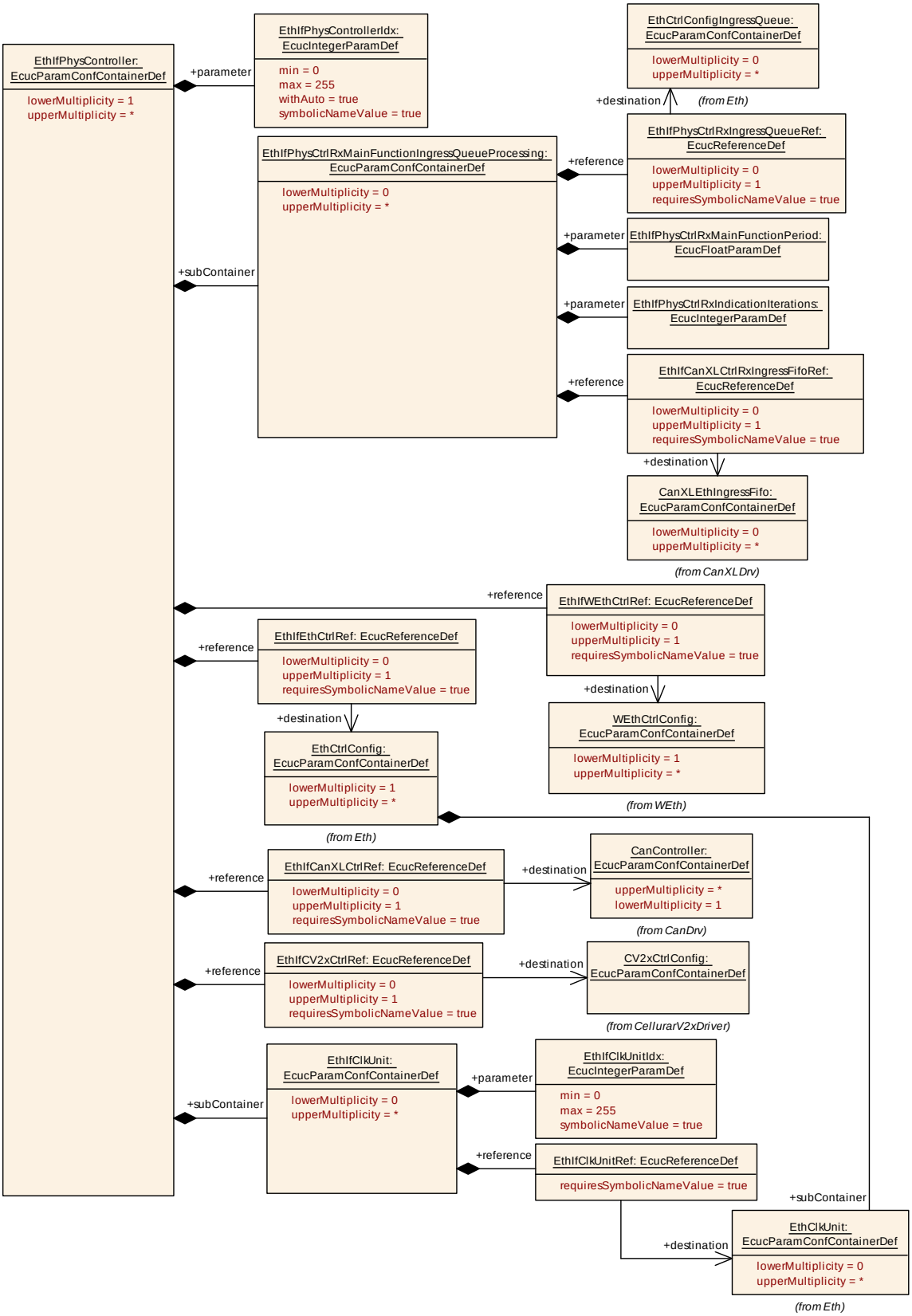

Figure 10.7: Ethernet Interface physical controller configuration structure

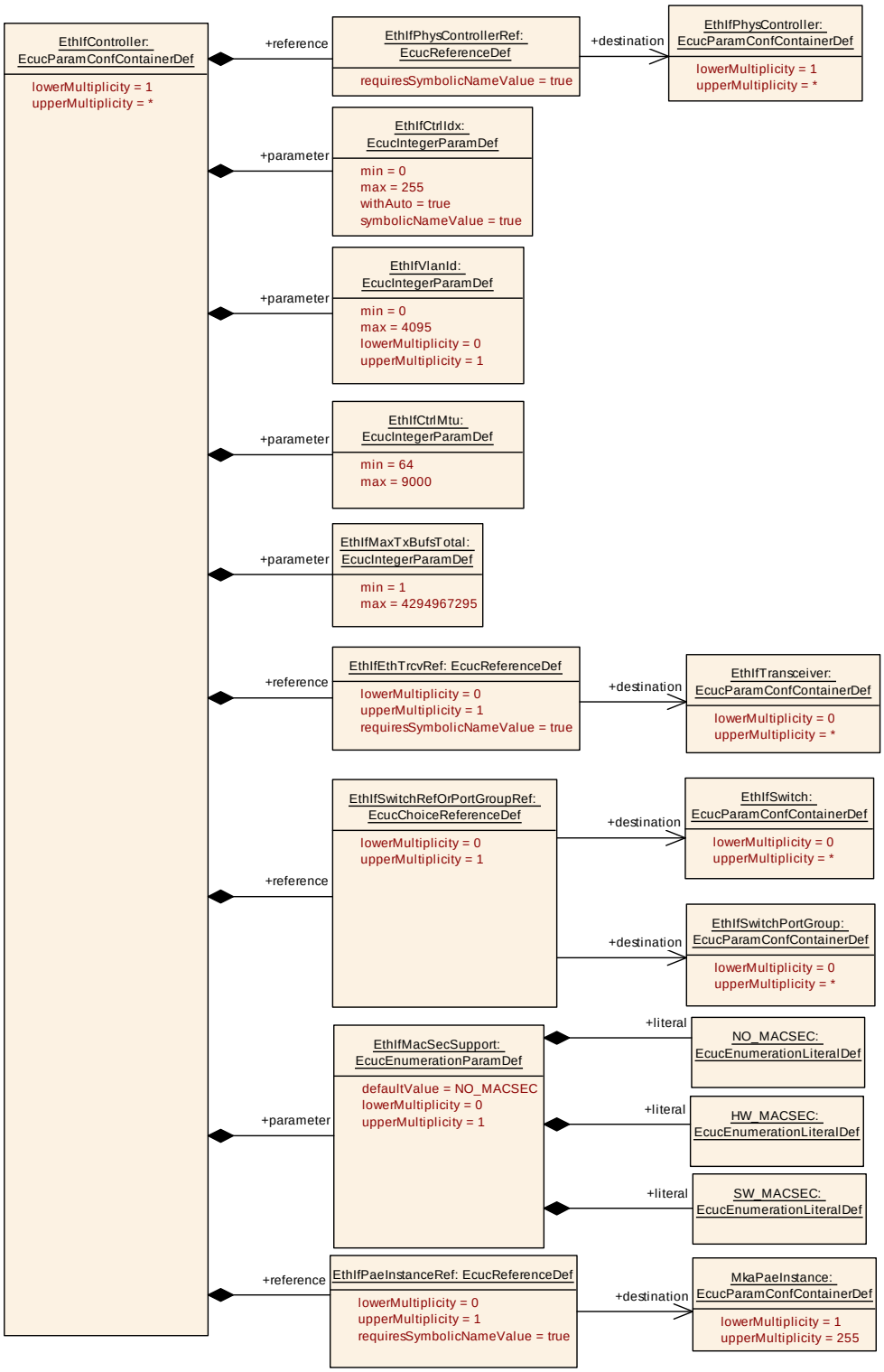

Figure 10.8: Ethernet Interface controller configuration structure

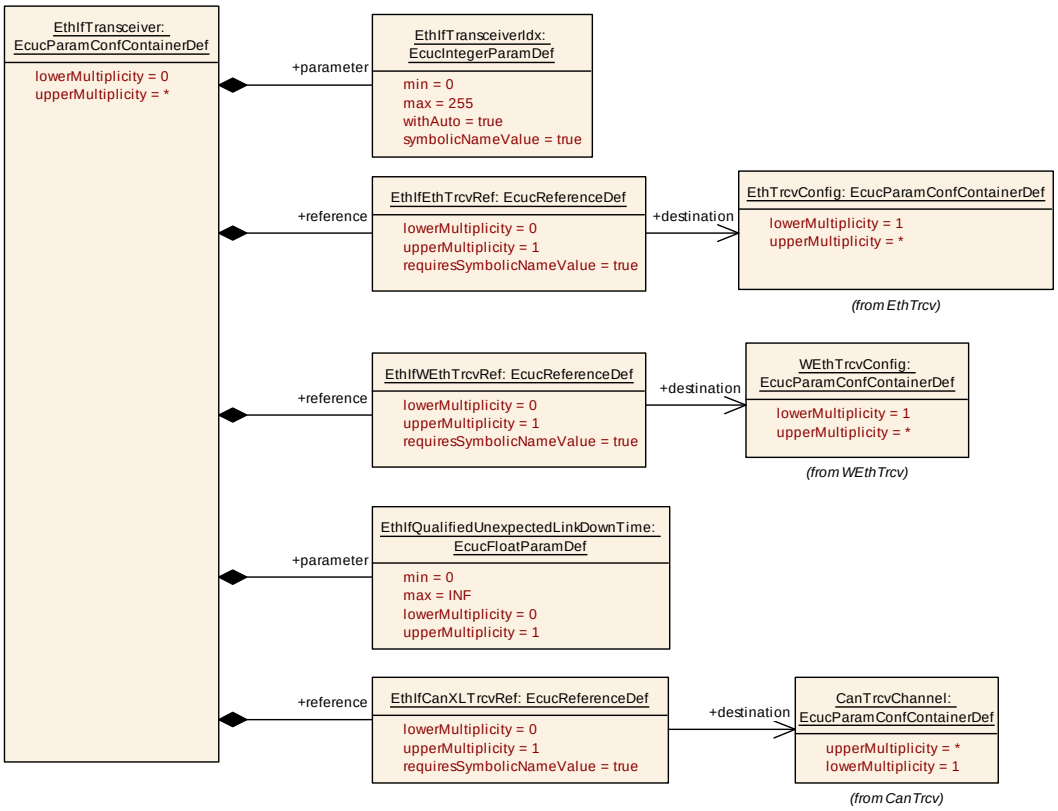

Figure 10.9: Ethernet Interface transceiver configuration structure

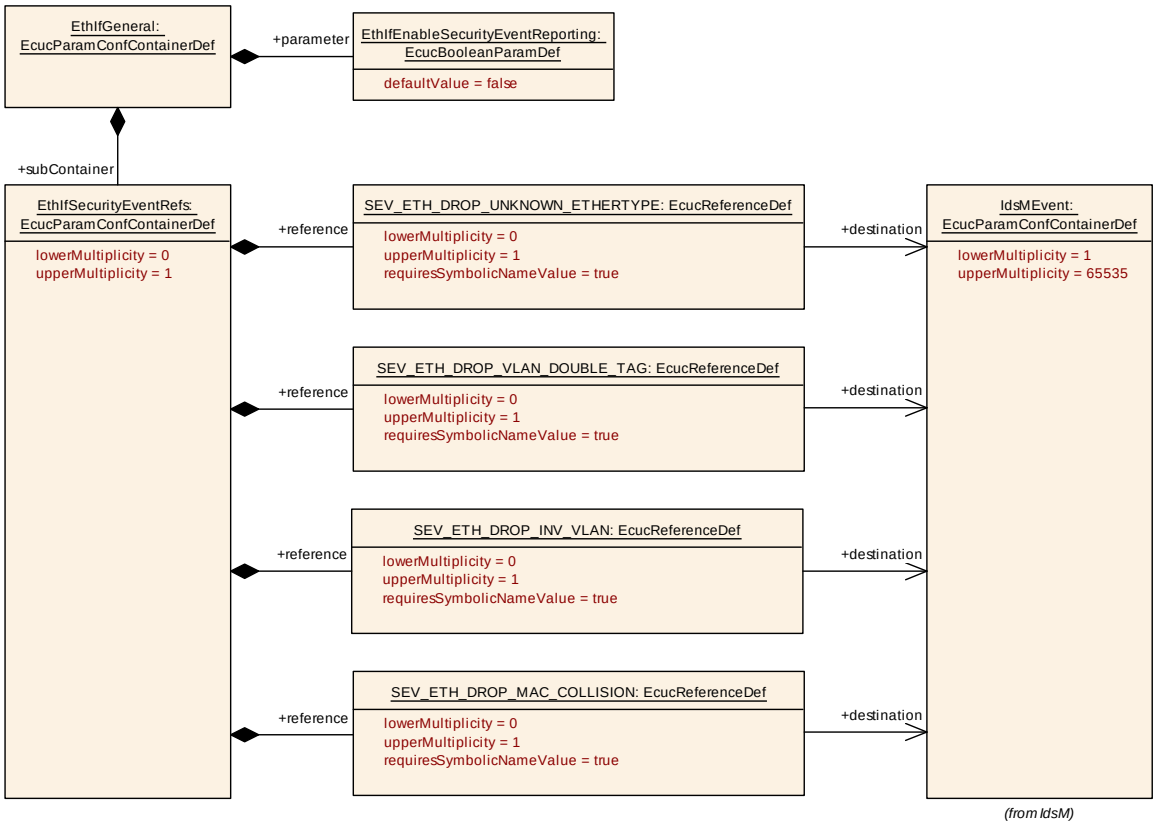

Figure 10.10: Ethernet security event ref

10.2.1 EthIf

SWS Item	[ECUC_EthIf_00049]
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	Scope / 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

SWS Item	[ECUC_EthIf_00001]
Container Name	EthIfGeneral
Parent Container	EthIf
Description	This container contains the general configuration parameters of the Ethernet Interface.
Configuration Parameters	

SWS Item	[ECUC_EthIf_00004]		
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	–	
Scope / Dependency	scope: local		

SWS Item	[ECUC_EthIf_00091]
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	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_EthIf_00005]**		
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	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_EthIf_00079]**		
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	–	
Scope / Dependency	scope: ECU		

SWS Item	**[ECUC_EthIf_00076]**		
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	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_EthIf_00006]**		
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	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_EthIf_00075]**		
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	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_EthIf_00094]**	
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	–	
Scope / Dependency	scope: ECU		

SWS Item	**[ECUC_EthIf_00072]**		
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	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_EthIf_00034]**		
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	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_EthIf_00035]**		
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	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_EthIf_00070]**		
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	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_EthIf_00071]**		
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	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_EthIf_00039]**		
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	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_EthIf_00023]**
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	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_EthIf_00056]**		
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	–	
Scope / Dependency	scope: local dependency: If parameter is defined, then EthIf_MainFunctionState shall be generated.		

SWS Item	**[ECUC_EthIf_00003]**		
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	–	
Scope / Dependency	scope: local		

SWS Item	[ECUC_EthIf_00102]		
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	–	
Scope / Dependency	scope: local		

SWS Item	[ECUC_EthIf_00055]		
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	–	
Scope / Dependency	scope: local		

SWS Item	[ECUC_EthIf_00024]
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	–	
Scope / Dependency	scope: ECU		

SWS Item	**[ECUC_EthIf_00030]**		
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	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_EthIf_00062]**		
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	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_EthIf_00077]**		
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	–	
Scope / Dependency	scope: local dependency: If this parameter is defined, the EthIf_MainFunctionState shall be generated and parameter EthIfEnableSignalQualityApi shall be set to TRUE.		

SWS Item	**[ECUC_EthIf_00033]**		
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	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_EthIf_00064]**		
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	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_EthIf_00054]**	
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	–	
Scope / Dependency	scope: local dependency: EthIfSwitchOffPortTimeDelay > (UdpNmTimeoutTime + UdpNmWaitBus SleepTime)		

SWS Item	**[ECUC_EthIf_00009]**		
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	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_EthIf_00063]**		
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	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_EthIf_00007]**
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	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_EthIf_00008]**		
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	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_EthIf_00040]**		
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	–	
Scope / Dependency	scope: local		

Included Containers		
Container Name	**Multiplicity**	**Scope / 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

10.2.3 EthIfConfigSet

SWS Item	[ECUC_EthIf_00010]
Container Name	EthIfConfigSet
Parent Container	EthIf
Description	Collecting container for all parameters with post-build configuration classes.
Configuration Parameters	

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
EthIfController	1..*	This container contains the configuration of EthIfController.
EthIfFrameOwnerConfig	1..*	Configuration of Ethernet frame owner
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

SWS Item	[ECUC_EthIf_00025]
Container Name	EthIfController
Parent Container	EthIfConfigSet
Description	This container contains the configuration of EthIfController.
Configuration Parameters	

SWS Item	[ECUC_EthIf_00026]
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	–	
Scope / Dependency	scope: ECU withAuto = true		

SWS Item	**[ECUC_EthIf_00032]**		
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
Scope / Dependency	scope: ECU 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).		

SWS Item	**[ECUC_EthIf_00089]**		
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
Scope / Dependency	scope: local		

SWS Item	**[ECUC_EthIf_00002]**		
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	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_EthIf_00029]**		
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
Scope / Dependency	scope: ECU		

SWS Item	**[ECUC_EthIf_00028]**
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
Scope / Dependency	scope: ECU		

SWS Item	**[ECUC_EthIf_00090]**		
Parameter Name	EthIfPaeInstanceRef		
Parent Container	EthIfController		
Description	Reference to MkaPaeInstance **Tags:** atp.Status=draft		
Multiplicity	0..1		
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
Scope / Dependency	scope: local		

SWS Item	**[ECUC_EthIf_00027]**		
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
Scope / Dependency	scope: ECU dependency: If EthIfEthTrcvRef is configured for an EthIfController then all EthIf Controller which refer to the same physical controller via EthIfPhysControllerRef (ECUC_EthIf_00027) shall reference the same EthIfTrcv via EthIfEthTrcvRef.		

SWS Item	**[ECUC_EthIf_00048]**
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
Scope / Dependency	scope: local dependency: * The configuration of EthIfSwitchRefOrPortGroupRef shall only be valid, if this EthIfController 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).		

No Included Containers

10.2.5 EthIfFrameOwnerConfig

SWS Item	**[ECUC_EthIf_00011]**
Container Name	EthIfFrameOwnerConfig
Parent Container	EthIfConfigSet
Description	Configuration of Ethernet frame owner
Configuration Parameters	

SWS Item	**[ECUC_EthIf_00012]**	
Parameter Name	EthIfFrameType	
Parent Container	EthIfFrameOwnerConfig	
Description	Selects the Ethernet frame type.	
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
Scope / Dependency	scope: local		

SWS Item	**[ECUC_EthIf_00013]**		
Parameter Name	EthIfOwner		
Parent Container	EthIfFrameOwnerConfig		
Description	Selects the owner of an Ethernet frame type. The owner is a zero based index into the callback function configuration 'EthIfRxIndicationConfig'. I.e. an Ethernet frame of type IPv4 (0x800) at index 0 will call the first callback function configured in 'EthIfRx IndicationConfig'.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
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
Scope / Dependency	scope: local		

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
EthIfFrameOwnerPdu	0..*	This container optionally defines the EcuC Pdu representing the specific EthIfFrameType Pdus for the upper layer module. Several container instances can be defined due to the following criteria: • Tx and Rx PDU definition • Optionally different EthIfPhysController relation This definition is only required if the upper layer module uses a Pdu based API (e.g. LSduR). **Tags:** atp.Status=draft

10.2.5.1 EthIfFrameOwnerPdu

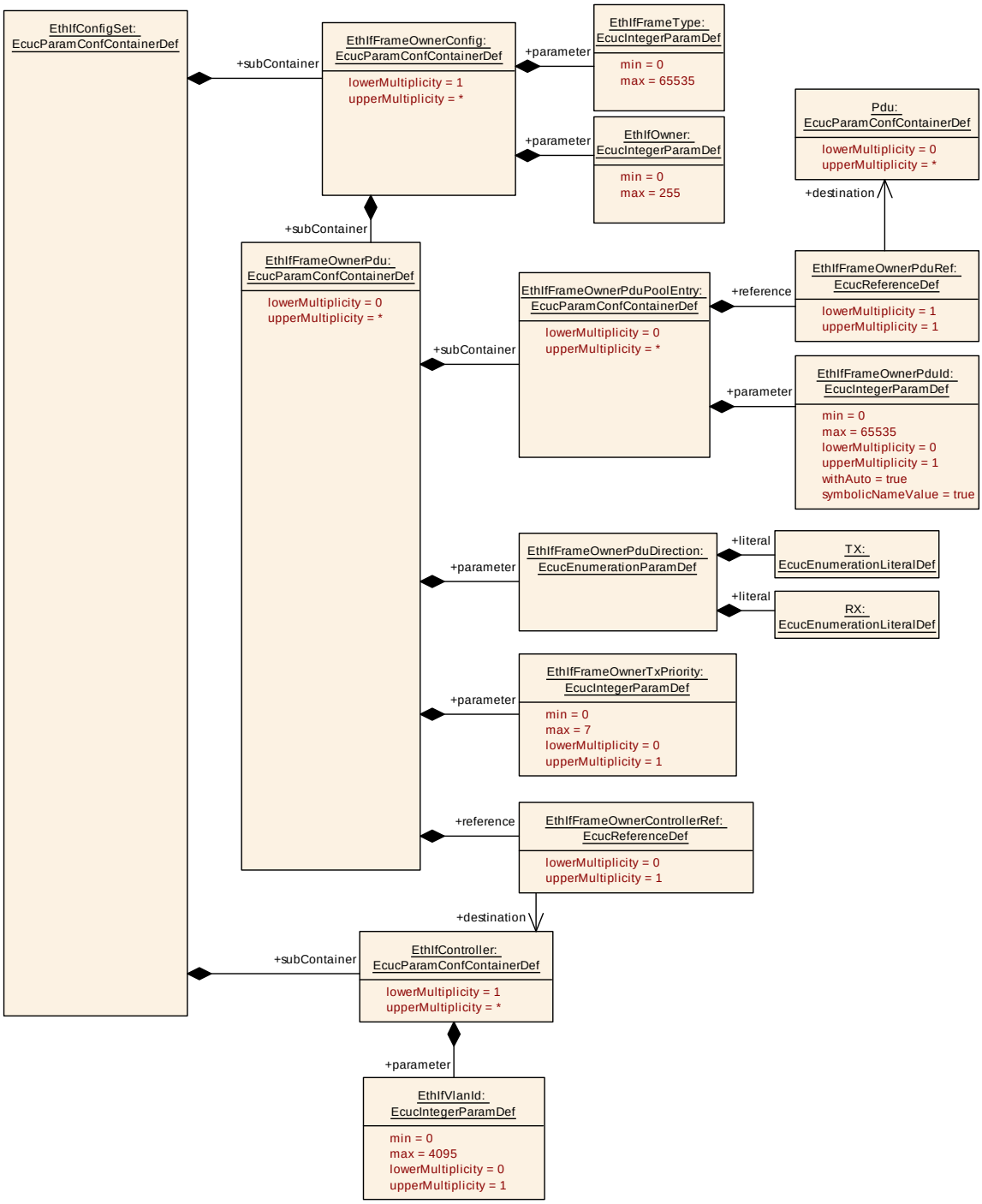

Figure 10.11: EthIfFrameOwnerPdu

In order to configure the Pdu based APIs for specific EtherTypes the container `EthIfFrameOwnerPdu` has been added to the `EthIfFrameOwnerConfig`. This is an optional extensions and allows to associate Frames of a specifc EtherType (`EthIfFrameType`) to a dedicated Pdu.

The parameter EthIfFrameOwnerPduDirection defines the direction of the respective Pdu assignment.EthIfFrameOwnerPduDirection defines the direction of the respective Pdu assignment.

The reference EthIfFrameOwnerControllerRef optionally defines a dedicated EthIfController to be considered in the evaluation.

If no EthIfFrameOwnerControllerRef is defined, then all messages matching the EthIfFrameType will be forwarded via that one Pdu defined in EthIfFrameOwnerPduRef. With this approach all messages, regardless of their origin network or VLAN, are uniformly handled by one Pdu (pair if EthIfFrameOwnerPduDirection is considered).

If a EthIfFrameOwnerControllerRef is defined, then only messages which match that referenced EthIfController are handled by that one Pdu (pair if EthIfFrameOwnerPduDirection is considered). With this approach also VLAN specific handling of EthIfFrameTypes can be defined.

SWS Item	[ECUC_EthIf_00095]		
Container Name	EthIfFrameOwnerPdu		
Parent Container	EthIfFrameOwnerConfig		
Description	This container optionally defines the EcuC Pdu representing the specific EthIfFrame Type Pdus for the upper layer module. Several container instances can be defined due to the following criteria: • Tx and Rx PDU definition • Optionally different EthIfPhysController relation This definition is only required if the upper layer module uses a Pdu based API (e.g. LSduR). **Tags:** atp.Status=draft		
Post-Build Variant Multiplicity	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

SWS Item	[ECUC_EthIf_00099]	
Parameter Name	EthIfFrameOwnerPduDirection	
Parent Container	EthIfFrameOwnerPdu	
Description	Defines whether the Pdu is received or transmitted. **Tags:** atp.Status=draft	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	RX	Frame is received with the defined EthIfFrame Type and forwarded to the upper layer as Pdu. **Tags:** atp.Status=draft

▽

△

	TX	Pdu is handed over from the upper layer and transmitted as a frame with the defined EthIf FrameType. **Tags:** atp.Status=draft	
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	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_EthIf_00100]**		
Parameter Name	EthIfFrameOwnerTxPriority		
Parent Container	EthIfFrameOwnerPdu		
Description	Definition of the VLAN Priority to be used for transmission. **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	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_EthIf_00101]**		
Parameter Name	EthIfFrameOwnerControllerRef		
Parent Container	EthIfFrameOwnerPdu		
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
Scope / Dependency	scope: local		

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
EthIfFrameOwnerPduPoolEntry	0..*	The container defines one entry in the EthIfFrameOwnerPduPool of Pdus to be used for the transport of a dedicated EthIfFrame OwnerPdu frame. Supported MetaData entry Rx: • TIMETUPLE_TYPE_PTR Supported MetaData entries Tx: • ETHERNET_MAC_64 • TIMETUPLE_TYPE_PTR • LISTELEM_PTR **Tags:** atp.Status=draft

SWS Item	**[ECUC_EthIf_00096]**		
Container Name	EthIfFrameOwnerPduPoolEntry		
Parent Container	EthIfFrameOwnerPdu		
Description	The container defines one entry in the EthIfFrameOwnerPduPool of Pdus to be used for the transport of a dedicated EthIfFrameOwnerPdu frame. Supported MetaData entry Rx: • TIMETUPLE_TYPE_PTR Supported MetaData entries Tx: • ETHERNET_MAC_64 • TIMETUPLE_TYPE_PTR • LISTELEM_PTR **Tags:** atp.Status=draft		
Post-Build Variant Multiplicity	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

SWS Item	**[ECUC_EthIf_00098]**	
Parameter Name	EthIfFrameOwnerPduId	
Parent Container	EthIfFrameOwnerPduPoolEntry	
Description	Definition of the Handle Pdu Id representing the specific EthIfFrameType Pdu for the upper layer module. **Tags:** atp.Status=draft	
Multiplicity	0..1	
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)	
Range	0 .. 65535	
Default value	–	

▽

△

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
Scope / Dependency	scope: ECU withAuto = true		

SWS Item	**[ECUC_EthIf_00097]**		
Parameter Name	EthIfFrameOwnerPduRef		
Parent Container	EthIfFrameOwnerPduPoolEntry		
Description	Reference to the EcuC Pdu representing the specific EthIfFrameType Pdu for 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
Scope / Dependency	scope: ECU		

No Included Containers

10.2.6 EthIfPhysController

SWS Item	**[ECUC_EthIf_00045]**
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.
Post-Build Variant Multiplicity	false
Configuration Parameters	

SWS Item	**[ECUC_EthIf_00046]**
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	–	
Scope / Dependency	scope: ECU withAuto = true		

SWS Item	**[ECUC_EthIf_00085]**		
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	–	
Scope / Dependency	scope: ECU dependency: The referenced CanController has to contain a CanXLController.		

SWS Item	**[ECUC_EthIf_00093]**		
Parameter Name	EthIfCV2xCtrlRef		
Parent Container	EthIfPhysController		
Description	Reference to physical Cellular V2X controller, which is handled by a specific Cellular V2X 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
Scope / Dependency	scope: ECU		

SWS Item	[ECUC_EthIf_00047]		
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
Scope / Dependency	scope: ECU		

SWS Item	[ECUC_EthIf_00073]		
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
Scope / Dependency	scope: ECU		

Included Containers		
Container Name	Multiplicity	Scope / 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

[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`.⌋*()*

10.2.7 EthIfPhysCtrlRxMainFunctionIngressQueueProcessing

SWS Item	[ECUC_EthIf_00106]		
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 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		
Post-Build Variant Multiplicity	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

SWS Item	[ECUC_EthIf_00052]		
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	–	
Scope / Dependency	scope: local		

SWS Item	[ECUC_EthIf_00051]		
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	–	
Scope / Dependency	scope: local		

SWS Item	[ECUC_EthIf_00087]		
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	–	
Scope / Dependency	scope: local dependency: Mutually exclusive with EthIfPhysCtrlRxIngressQueueRef. One of the two parameters is required.		

SWS Item	[ECUC_EthIf_00053] (Obsolete)		
Parameter Name	EthIfPhysCtrlRxIngressFifoRef		
Parent Container	EthIfPhysCtrlRxMainFunctionIngressQueueProcessing		
Description	Reference to the reception FIFO. **Tags:** atp.Status=obsolete		
Multiplicity	0..1		
Type	Symbolic name reference to EthCtrlConfigIngressFifo		
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	–	
Scope / Dependency	scope: local dependency: Mutually exclusive with EthIfCanXLCtrlRxIngressFifoRef. One of the two parameters is required.		

SWS Item	[ECUC_EthIf_00107]		
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	–	
Scope / Dependency	scope: local dependency: Mutually exclusive with EthIfCanXLCtrlRxIngressFifoRef. One of the two parameters is required.		

No Included Containers

10.2.8 EthIfClkUnit

SWS Item	**[ECUC_EthIf_00105]**		
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		
Post-Build Variant Multiplicity	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

SWS Item	**[ECUC_EthIf_00104]**		
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	–	
Scope / Dependency	scope: ECU		

SWS Item	**[ECUC_EthIf_00103]**
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	–	
Scope / Dependency	scope: ECU		

No Included Containers

10.2.9 EthIfRxIndicationConfig

SWS Item	**[ECUC_EthIf_00014]**
Container Name	EthIfRxIndicationConfig
Parent Container	EthIfConfigSet
Description	Configuration of receive callback functions.
Configuration Parameters	

SWS Item	**[ECUC_EthIf_00015]**		
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
Scope / Dependency	scope: local		

No Included Containers

10.2.10 EthIfSwitch

SWS Item	[ECUC_EthIf_00036]
Container Name	EthIfSwitch
Parent Container	EthIfConfigSet
Description	This container contains the configuration of EthIfSwitches.
Configuration Parameters	

SWS Item	[ECUC_EthIf_00037]		
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	–	
Scope / Dependency	scope: ECU withAuto = true		

SWS Item	[ECUC_EthIf_00038]		
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
Scope / Dependency	scope: ECU		

No Included Containers

10.2.11 EthIfSwitchPortGroup

SWS Item	[ECUC_EthIf_00057]
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.
Configuration Parameters	

SWS Item	**[ECUC_EthIf_00058]**		
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	–	
Scope / Dependency	scope: ECU withAuto = true		

SWS Item	**[ECUC_EthIf_00059]**		
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
Scope / Dependency	scope: local dependency: only valid if a EthIfSwitchRefOrPortGroupRef refers to the EthIfSwitch PortGroup.		

SWS Item	**[ECUC_EthIf_00060]**
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
Scope / Dependency	scope: local		

No Included Containers

10.2.12 EthIfTransceiver

SWS Item	**[ECUC_EthIf_00042]**
Container Name	EthIfTransceiver
Parent Container	EthIfConfigSet
Description	This container contains the configuration of EthIfTransceiver. The usage of EthIfEthTrcvRef, EthIfCanXLTrcvRef, and EthIfWEthTrcvRef is exclusive OR.
Post-Build Variant Multiplicity	false
Configuration Parameters	

SWS Item	**[ECUC_EthIf_00078]**		
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
Scope / Dependency	scope: ECU dependency: 1.) If this parameter is set, EthIf_MainFunctionState has to be available 2.) Only applicable if the referenced EthTrcv has set EthTrcvActAsSlavePassive Enabled to TRUE.		

SWS Item	**[ECUC_EthIf_00043]**
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	–	
Scope / Dependency	scope: ECU withAuto = true		

SWS Item	**[ECUC_EthIf_00086]**		
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	–	
Scope / Dependency	scope: ECU dependency: The referenced CanTrcvChannel has to contain a CanTrcvXLChannel.		

SWS Item	**[ECUC_EthIf_00044]**		
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
Scope / Dependency	scope: ECU		

SWS Item	**[ECUC_EthIf_00074]**
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
Scope / Dependency	scope: ECU		

No Included Containers

10.2.13 EthIfTrcvLinkStateChgConfig

SWS Item	**[ECUC_EthIf_00018]**
Container Name	EthIfTrcvLinkStateChgConfig
Parent Container	EthIfConfigSet
Description	Specifies link state change callback function
Configuration Parameters	

SWS Item	**[ECUC_EthIf_00019]**		
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
Scope / Dependency	scope: local		

No Included Containers

10.2.14 EthIfTxConfirmationConfig

SWS Item	**[ECUC_EthIf_00016]**
Container Name	EthIfTxConfirmationConfig
Parent Container	EthIfConfigSet

▽

△

Description	Configuration of transmit indication callback functions.
Configuration Parameters	

SWS Item	**[ECUC_EthIf_00017]**		
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
Scope / Dependency	scope: local		

No Included Containers

10.2.15 EthIfSecurityEventRefs

SWS Item	**[ECUC_EthIf_00080]**		
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		
Post-Build Variant Multiplicity	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

SWS Item	**[ECUC_EthIf_00083]**
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	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_EthIf_00084]**		
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	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_EthIf_00081]**		
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	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_EthIf_00082]**
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	–	
Scope / Dependency	scope: local		

No Included Containers

10.3 Published Information

For details refer to the chapter 10.3 “Published Information” in SWS_BSWGeneral [6].

A Not applicable requirements

[SWS_EthIf_00999] ⌈These requirements are not applicable to this specification.⌋ *(SRS_BSW_00170)*

B Change History

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

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

B.1.1 Added Specification Items in R22-11

Number	Heading
[SWS_EthIf_00520]	
[SWS_EthIf_00521]	
[SWS_EthIf_00522]	
[SWS_EthIf_00523]	
[SWS_EthIf_00524]	
[SWS_EthIf_00525]	
[SWS_EthIf_00526]	
[SWS_EthIf_00531]	
[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]	

△

Number	Heading
[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]	
[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]	

▽

△

Number	Heading
[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]	
[SWS_EthIf_91233]	
[SWS_EthIf_91234]	
[SWS_EthIf_91235]	
[SWS_EthIf_91236]	
[SWS_EthIf_91237]	
[SWS_EthIf_91238]	
[SWS_EthIf_-CONSTR_00001]	

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

B.1.2 Changed Specification Items in R22-11

Number	Heading
[SWS_EthIf_00017]	
[SWS_EthIf_00023]	
[SWS_EthIf_00024]	
[SWS_EthIf_00034]	
[SWS_EthIf_00035]	

▽

△

Number	Heading
[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]	
[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]	

▽

△

Number	Heading
[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]	
[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]	

▽

△

Number	Heading
[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]	
[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.2: Changed Specification Items in R22-11

B.1.3 Deleted Specification Items in R22-11

none

B.1.4 Added Constraints in R22-11

none

B.1.5 Changed Constraints in R22-11

none

B.1.6 Deleted Constraints in R22-11

none

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

B.2.1 Added Specification Items in R23-11

Number	Heading
[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]	

▽

△

Number	Heading
[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]	
[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]	

▽

△

Number	Heading
[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
[SWS_EthIf_91139]	Definition of scheduled function EthIf_MainFunctionRx_<IngressQueue Processing ShortName>

Table B.3: Added Specification Items in R23-11

B.2.2 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

▽

△

Number	Heading
[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.4: Changed Specification Items in R23-11

B.2.3 Deleted Specification Items in R23-11

none

B.2.4 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.5: Added Constraints in R23-11

B.2.5 Changed Constraints in R23-11

none

B.2.6 Deleted Constraints in R23-11

none