Document Title	Specification of Ethernet Switch Driver
Document Category	Software specification (SWS)
Document UID	656
Document Status	published
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	R26-11

Document Change History			
Date	**Release**	**Changed by**	**Description**
2026-11-26	R26-11	AUTOSAR Release Management	• Further revised MacSec support • Eth_DataType, GetVersionInfo() removed • Bugfixes (types, VLAN handling, etc.)
2025-11-27	R25-11	AUTOSAR Release Management	• Abstraction from Driver APIs through EthIf • Revised MacSec support • Bugfixes (types, VLAN handling, etc.)
2024-11-27	R24-11	AUTOSAR Release Management	• Added shared and independent VLAN learning modes • Added frame preemption support • Added MMD support • Added multi-gigabit interface support
2023-11-23	R23-11	AUTOSAR Release Management	• Concept 710 (Deterministic communication with TSN) incorporated • Interaction with the Firewall module added
2022-11-24	R22-11	AUTOSAR Release Management	• Remodel EthSwtPort MAC address and VLAN definition • Implemented MACSec • Implemented Derterministic Communication with TSN
2021-11-25	R21-11	AUTOSAR Release Management	• Added 10BASE-T1S support • Clarified return values and development errors • Removed EthSwtMgmtInfoIndicationTimeout related requirements

△

2020-11-30	R20-11	AUTOSAR Release Management	• Support for Ethernet wake on data line • Modified description of return values in EthSwt_StoreConfiguration, EthSwt_ResetConfiguration • Updated the types for ReTaggingVlanId and DoubleTaggingVlanId • Fix service IDs for EthSwt_DeletePortMirrorConfiguration, EthSwt_PortLinkStateRequest, EthSwt_GetMaxFIFOBufferFillLevel • Editorial changes
2019-11-28	R19-11	AUTOSAR Release Management	• Possibility to explicitly request or release Ethernet link state added • Replace usage of EthTrcv_ModeType with the Eth_ModeType • Support for 2500 MBit/s Ethernet connection • Fix Ethernet Hardware Initialization • Changed Document Status from Final to published
2018-10-31	4.4.0	AUTOSAR Release Management	• Clarified Port Mirroring concepts. • Introduced timeout for ARL table entries • Added counter synchronization for cascaded switches
2017-12-08	4.3.1	AUTOSAR Release Management	• minor corrections / clarifications / editorial changes; For details please refer to the ChangeDocumentation
2016-11-30	4.3.0	AUTOSAR Release Management	• Restructured VLAN-membership as a port-related configuration parameter • Introduced configuration of rate policers on ingress side • Introduced filter configuration for double tagged frames • Introduced configuration of minimum buffer size for FIFOS • Introduced Types to read HW- statistic by List pointer; reorganized interfaces to read HW-statistics.

▽

▽

△

			△ • Introduced Compensation of Ethernet switch delays for Global Time Synchronization • Add / update elements to describe MAC interface and physical interface • Added testing functionality for diagnostic use cases • Added Possibility to switch off ports and switch instances according to VLAN or PNC. • Introduced interfaces for verification of switch configuration
2015-07-31	4.2.2	AUTOSAR Release Management	• minor corrections / clarifications / editorial changes; For details please refer to the ChangeDocumentation
2014-10-31	4.2.1	AUTOSAR Release Management	• Initial Release

Disclaimer

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

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

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

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

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

Table of Contents

1 Introduction and functional overview 12

2 Acronyms, abbreviations and definitons 13
2.1 Acronyms and abbreviations . . . 13
2.2 Definitions . . . 13
2.2.1 Ethernet packet . . . 13
2.2.2 Ethernet frame . . . 13
2.2.3 Stream . . . 14
2.2.4 Stream identification . . . 14

3 Related documentation 15
3.1 Input documents & related standards and norms . . . 15
3.2 Related specification . . . 16

4 Constraints and assumptions 17
4.1 Constraints . . . 17
4.2 Assumptions . . . 18
4.3 Applicability to car domains . . . 19

5 Dependencies to other modules 20

6 Requirements Tracing 21

7 Functional specification 26
7.1 Ethernet BSW stack . . . 26
7.1.1 Indexing scheme . . . 27
7.1.2 Ethernet Switch Port Mirroring . . . 28
7.1.3 State Handling . . . 29
7.1.4 Handling of cable diagnostic . . . 29
7.1.5 Support of frame preemption . . . 30
7.1.6 Interaction with the Firewall module . . . 31
7.1.7 Functional Description . . . 34
7.1.7.1 Learning Phase at Start-up . . . 34
7.1.7.2 Frame forwarding process . . . 39
7.1.7.2.1 Ensure VLAN-ID and priority availability . . . 41
7.1.7.2.2 Priority-Code-Point-Regeneration . . . 44
7.1.7.2.3 Stream identification . . . 44
7.1.7.2.4 Ingress filtering . . . 51
7.1.7.2.5 Frame filtering . . . 56
7.1.7.2.6 Egress filtering . . . 58
7.1.7.2.7 Per-stream filtering and policing . . . 58
7.1.7.2.8 Queueing frames . . . 64
7.1.7.2.9 Shapers and transmission selection . . . 68
7.1.7.2.10 Transmission on the network . . . 79

7.1.7.3 Switch Management support . . . 79
7.1.7.4 Global Time support . . . 81
7.1.7.5 Counter synchronization of Ethernet switches which are connected via uplink ports . . . 81
7.1.7.6 Verification of Configuration . . . 82
7.1.7.7 Testing and Diagnostic of Switch Ports . . . 82
7.1.7.8 Low Power Mode Support . . . 83
7.1.7.9 MACsec support . . . 83
7.2 Error Classifications . . . 97
7.2.1 Development Errors . . . 98
7.2.2 Runtime Errors . . . 100
7.2.3 Transient Faults . . . 100
7.2.4 Production Errors . . . 101
8 API specification 102
8.1 Imported types . . . 102
8.2 Type definitions . . . 103
8.2.1 EthSwt_StateType . . . 103
8.2.2 EthSwt_ConfigType . . . 103
8.2.3 EthSwt_MacLearningType . . . 104
8.2.4 EthSwt_MgmtInfoType . . . 104
8.2.5 EthSwt_PortMirrorCfgType . . . 105
8.2.6 EthSwt_PortMirrorStateType . . . 106
8.2.7 EthSwt_ReturnType . . . 107
8.2.8 EthSwt_MgmtOwner . . . 107
8.2.9 EthSwt_Mgmt_ObjectType . . . 107
8.2.10 EthSwt_MgmtObjectValidType . . . 108
8.3 Function definitions . . . 109
8.3.1 Generic functions . . . 109
8.3.1.1 EthSwt_Init . . . 109
8.3.1.2 EthSwt_GetSwitchIdentifier . . . 111
8.3.1.3 EthSwt_SetForwardingMode . . . 112
8.3.1.4 EthSwt_CheckWakeup . . . 112
8.3.2 Switch Configuration . . . 113
8.3.2.1 EthSwt_VerifyConfig . . . 113
8.3.2.2 EthSwt_StoreConfiguration . . . 114
8.3.2.3 EthSwt_ResetConfiguration . . . 115
8.3.2.4 EthSwt_NvmSingleBlockCallback . . . 115
8.3.2.5 EthSwtPersistentConfigurationResultCallback . . . 117
8.3.2.6 EthSwt_GetCfgDataRaw . . . 117
8.3.2.7 EthSwt_GetCfgDataInfo . . . 118
8.3.3 Port Mode . . . 119
8.3.3.1 EthSwt_SetSwitchPortMode . . . 119
8.3.3.2 EthSwt_GetSwitchPortMode . . . 121

8.3.3.3 EthSwt_StartSwitchPortAutoNegotiation . . . 122
8.3.3.4 EthSwt_GetSwitchPortWakeupReason . . . 123
8.3.3.5 EthSwt_GetLinkState . . . 124
8.3.3.6 EthSwt_GetBaudRate . . . 125
8.3.3.7 EthSwt_GetDuplexMode . . . 126
8.3.3.8 EthSwt_GetMaxQueueBufferFillLevel . . . 127
8.3.3.9 EthSwt_GetPortIdentifier . . . 128
8.3.3.10 EthSwt_PortEnableTimeStamp . . . 129
8.3.3.11 EthSwt_GetPortSignalQuality . . . 130
8.3.4 MAC address and VLAN related . . . 131
8.3.4.1 EthSwt_GetPortMacAddr . . . 131
8.3.4.2 EthSwt_GetPortMacAddrVlan . . . 132
8.3.4.3 EthSwt_GetArlTable . . . 134
8.3.4.4 EthSwt_EnableVlan . . . 135
8.3.4.5 EthSwt_SetMacLearningMode . . . 136
8.3.4.6 EthSwt_GetMacLearningMode . . . 137
8.3.5 MIB counter access . . . 138
8.3.5.1 EthSwt_GetCounterValues . . . 138
8.3.5.2 EthSwt_GetRxStats . . . 139
8.3.5.3 EthSwt_GetTxStats . . . 140
8.3.5.4 EthSwt_GetTxErrorCounterValues . . . 141
8.3.6 Switch and transceiver management . . . 142
8.3.6.1 EthSwt_GetSwitchReg . . . 142
8.3.6.2 EthSwt_SetSwitchReg . . . 143
8.3.6.3 EthSwt_ReadTrcvRegister . . . 144
8.3.6.4 EthSwt_WriteTrcvRegister . . . 145
8.3.6.5 EthSwt_ReadMmd . . . 146
8.3.6.6 EthSwt_WriteMmd . . . 146
8.3.7 Frame Handling . . . 147
8.3.7.1 EthSwt_EthRxProcessFrame . . . 147
8.3.7.2 EthSwt_EthRxFinishedIndication . . . 148
8.3.7.3 EthSwt_EthTxPrepareFrame . . . 149
8.3.7.4 EthSwt_EthTxAdaptBufferLength . . . 150
8.3.7.5 EthSwt_EthTxProcessFrame . . . 151
8.3.7.6 EthSwt_EthTxFinishedIndication . . . 152
8.3.8 Port Mirroring . . . 152
8.3.8.1 EthSwt_WritePortMirrorConfiguration . . . 152
8.3.8.2 EthSwt_ReadPortMirrorConfiguration . . . 154
8.3.8.3 EthSwt_DeletePortMirrorConfiguration . . . 155
8.3.8.4 EthSwt_GetPortMirrorState . . . 156
8.3.8.5 EthSwt_SetPortMirrorState . . . 157
8.3.9 Test modes and cable diagnostic . . . 158
8.3.9.1 EthSwt_SetPortTestMode . . . 158

8.3.9.2 EthSwt_SetPortLoopbackMode . . . 159
8.3.9.3 EthSwt_RunPortCableDiagnostic . . . 160
8.3.9.4 EthSwt_GetPortCableDiagnosticsResult . . . 160
8.3.9.5 EthSwt_SetPortTxMode . . . 161
8.3.9.6 EthSwt_PortLinkStateRequest . . . 162
8.3.10 Switch Management . . . 163
8.3.10.1 EthSwt_GetRxMgmtObject . . . 163
8.3.10.2 EthSwt_GetTxMgmtObject . . . 164
8.3.10.3 EthSwt_SetMgmtInfo . . . 164
8.3.11 MacSec API . . . 165
8.3.11.1 EthSwt_MacSecUpdateSecY . . . 165
8.3.11.2 EthSwt_MacSecInitRxSc . . . 166
8.3.11.3 EthSwt_MacSecResetRxSc . . . 166
8.3.11.4 EthSwt_MacSecAddTxSa . . . 167
8.3.11.5 EthSwt_MacSecUpdateTxSa . . . 168
8.3.11.6 EthSwt_MacSecDeleteTxSa . . . 169
8.3.11.7 EthSwt_MacSecAddRxSa . . . 169
8.3.11.8 EthSwt_MacSecUpdateRxSa . . . 170
8.3.11.9 EthSwt_MacSecDeleteRxSa . . . 171
8.3.11.10 EthSwt_MacSecGetTxSaNextPn . . . 172
8.3.11.11 EthSwt_MacSecGetMacSecStatistics . . . 172
8.3.11.12 EthSwt_MacSecSetControlledPortEnabled . . . 173
8.3.12 Streams . . . 174
8.3.12.1 EthSwt_ExtractStreamHandleIdx . . . 174
8.3.12.2 EthSwt_GetStreamStatistics . . . 174
8.3.12.3 EthSwt_SetStreamState . . . 175
8.4 Scheduled functions . . . 176
8.4.1 EthSwt_MainFunction . . . 176
8.4.2 EthSwt_BackgroundTask . . . 176
8.5 Expected interfaces . . . 176
8.5.1 Mandatory Interfaces . . . 177
8.5.2 Optional Interfaces . . . 177
8.5.3 Configurable interfaces . . . 178
8.5.3.1 <EthSwtLinkDownCallout> . . . 179
8.5.3.2 <EthSwtLinkUpCallout> . . . 179
8.5.3.3 <GetCfgDataRawDone> . . . 180
8.6 Service Interfaces . . . 180
9 Sequence diagrams 181
9.1 Switch Management support . . . 181
10 Configuration specification 184
10.1 Containers and configuration parameters . . . 184
10.1.1 EthSwt . . . 184

10.1.2 EthSwtGeneral . . . 186
10.1.3 EthSwtConfig . . . 207
10.1.4 EthSwtAtsGroupInstanceTable . . . 212
10.1.5 EthSwtAtsGroupInstanceEntry . . . 212
10.1.6 EthSwtDemEventParameterRefs . . . 213
10.1.7 EthSwtMacForwardingTable . . . 214
10.1.8 EthSwtNvm . . . 216
10.1.9 EthSwtPSCM . . . 217
10.1.10 EthSwtAtsInstanceTable . . . 218
10.1.11 EthSwtAtsInstanceEntry . . . 219
10.1.12 EthSwtPSFP . . . 221
10.1.13 EthSwtFilterMaxSduSizeTable . . . 222
10.1.14 EthSwtFilterMaxSduSizeEntry . . . 223
10.1.15 EthSwtFlowMeteringTable . . . 224
10.1.16 EthSwtFlowMeteringEntry . . . 225
10.1.17 EthSwtStreamFilterTable . . . 228
10.1.18 EthSwtStreamFilterEntry . . . 229
10.1.19 EthSwtAssignedStreamHandle . . . 233
10.1.20 EthSwtStreamGateTable . . . 235
10.1.21 EthSwtStreamGateEntry . . . 236
10.1.22 EthSwtPort . . . 236
10.1.23 EthSwtPortEgress . . . 246
10.1.24 EthSwtPortPriorityToTrafficClassAssignment . . . 250
10.1.25 EthSwtPortEgressScheduler . . . 252
10.1.26 EthSwtPortEgressSchedulerPredecessor . . . 253
10.1.27 EthSwtPortQueue . . . 254
10.1.28 EthSwtPortEgressQueueTransmissionSelection . . . 256
10.1.29 EthSwtPortEgressQueueTransmissionSelectionCBSConfig . . . 258
10.1.30 EthSwtPortEgressQueueTransmissionSelectionETSConfig . . . 259
10.1.31 EthSwtPortIngress . . . 261
10.1.32 EthSwtPortIngressScheduler . . . 264
10.1.33 EthSwtPortIngressVlanTranslationTable . . . 264
10.1.34 EthSwtPortIngressVlanTranslationTableEntry . . . 265
10.1.35 EthSwtPortPriorityRegeneration . . . 266
10.1.36 EthSwtSpi . . . 267
10.1.37 EthSwtSpiSequence . . . 268
10.1.38 EthSwtStreamIdentificationTable . . . 270
10.1.39 EthSwtStreamIdentificationEntry . . . 272
10.1.40 EthSwtStreamFilterAction . . . 274
10.1.41 EthSwtStreamFilterActionDestinationPortModification . . . 276
10.1.42 EthSwtStreamFilterActionVlanModification . . . 277
10.1.43 EthSwtStreamFilterRule . . . 278
10.1.44 EthSwtStreamFilterIPDestAddress . . . 282

10.1.45 EthSwtStreamFilterIPSrcAddress . . . 283
10.1.46 EthSwtStreamFilterMACDestAddress . . . 284
10.1.47 EthSwtStreamFilterMACSrcAddress . . . 285
10.1.48 EthSwtStreamFilterTcpDestPort . . . 286
10.1.49 EthSwtStreamFilterTcpSrcPort . . . 287
10.1.50 EthSwtStreamFilterUdpDestPort . . . 288
10.1.51 EthSwtStreamFilterUdpSrcPort . . . 288
10.1.52 EthSwtStreamFilterIEEE1722StreamId . . . 289
10.1.53 EthSwtUnknownMacDestAddressConfig . . . 290
10.1.54 EthSwtVlanMembership . . . 292
10.1.55 EthSwtVlanMembershipPortRefEntry . . . 293
10.2 Constraints . . . 296
A Change History 297
A.1 Traceable item history of this document according to AUTOSAR Release R22-11 . . . 297
A.1.1 Added Specification Items in R22-11 . . . 297
A.1.2 Changed Specification Items in R22-11 . . . 299
A.1.3 Deleted Specification Items in R22-11 . . . 303
A.1.4 Added Constraints in R22-11 . . . 303
A.1.5 Changed Constraints in R22-11 . . . 303
A.1.6 Deleted Constraints in R22-11 . . . 303
A.2 Traceable item history of this document according to AUTOSAR Release R23-11 . . . 303
A.2.1 Added Specification Items in R23-11 . . . 303
A.2.2 Changed Specification Items in R23-11 . . . 304
A.2.3 Deleted Specification Items in R23-11 . . . 305
A.2.4 Added Constraints in R23-11 . . . 305
A.2.5 Changed Constraints in R23-11 . . . 306
A.2.6 Deleted Constraints in R23-11 . . . 306
A.3 Traceable item history of this document according to AUTOSAR Release R24-11 . . . 307
A.3.1 Added Specification Items in R24-11 . . . 307
A.3.2 Changed Specification Items in R24-11 . . . 310
A.3.3 Deleted Specification Items in R24-11 . . . 311
A.3.4 Added Constraints in R24-11 . . . 312
A.3.5 Changed Constraints in R24-11 . . . 313
A.3.6 Deleted Constraints in R24-11 . . . 313
A.4 Traceable item history of this document according to AUTOSAR Release R25-11 . . . 314
A.4.1 Added Specification Items in R25-11 . . . 314
A.4.2 Changed Specification Items in R25-11 . . . 317
A.4.3 Deleted Specification Items in R25-11 . . . 318
A.4.4 Added Constraints in R25-11 . . . 319

A.4.5 Changed Constraints in R25-11 319
A.4.6 Deleted Constraints in R25-11 319
A.5 Traceable item history of this document according to AUTOSAR Release R26-11 319
A.5.1 Added Specification Items in R26-11 319
A.5.2 Changed Specification Items in R26-11 320
A.5.3 Deleted Specification Items in R26-11 320
A.5.4 Added Constraints in R26-11 321
A.5.5 Changed Constraints in R26-11 321
A.5.6 Deleted Constraints in R26-11 321

1 Introduction and functional overview

In the AUTOSAR Layered Software Architecture [1], the Ethernet Switch Driver belongs to the Communication Hardware Abstraction.

This indicates the main task of the Ethernet Switch Driver:

Provide to the upper layers (e.g. Ethernet Interface [2]) a hardware independent interface comprising a switch with several ports. This interface shall be uniform for all Ethernet switches. Thus, the upper layers may access the underlying communication technology in a uniform manner.

A single Ethernet Switch Driver module supports only one type of switch hardware. The Ethernet physical layer ports are configured by the Ethernet Transceiver Driver[3]. The Ethernet Switch Driver's prefix generates a unique namespace. The Ethernet Interface can access different Ethernet controller types using different Ethernet Switch Drivers using this prefix. The decision which driver to use to access a particular transceiver is a configuration parameter of the Ethernet Interface.

Figure 1.1 depicts the lower part of the Ethernet stack. Accesses via an SPI- and MII/MDIO-Hardware-Interface for switch specific configuration or functions are done via the Ethernet Driver [4] or the SPI driver [5].

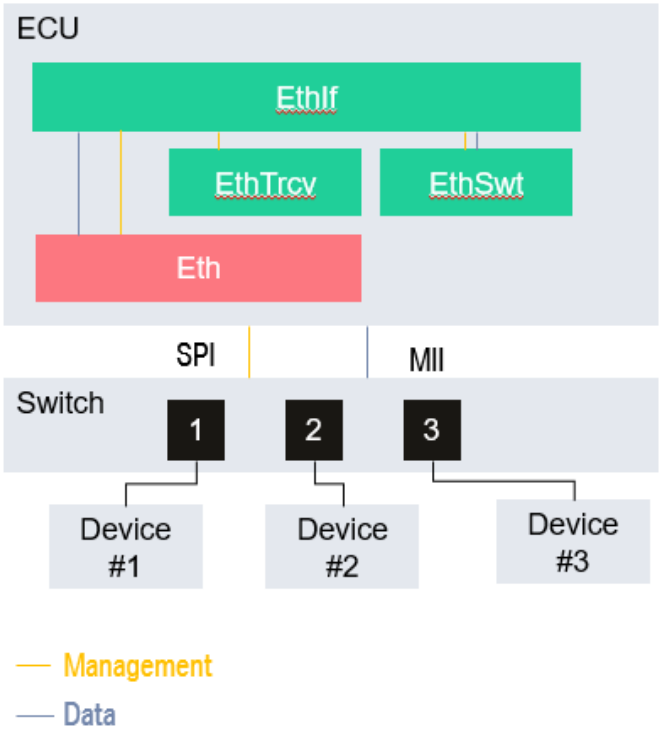

Figure 1.1: Ethernet Switch Driver in layer architecture

2 Acronyms, abbreviations and definitons

The glossary below includes acronyms, abbreviations and definitions relevant to the Ethernet Switch Driver module that are not included in the [6, AUTOSAR glossary].

2.1 Acronyms and abbreviations

Acronym / Abbreviation:	Description:
ARL	Address resolution table. The table contains the (VLAN-specific) MAC address to Ethernet switch port mapping, i.e., for SVL the MAC address to port mapping is learnt and stored as an ARL entry in the ARL table, while for IVL the combination of MAC address and VLAN is associated with a single ARL entry upon learning.
EcuM	ECU State Manager module
Eth	Ethernet Controller Driver (AUTOSAR BSW module)
EthIf	Ethernet Interface (AUTOSAR BSW module)
EthTrcv	Ethernet Transceiver Driver (AUTOSAR BSW module)
IVL	Independent VLAN learning defined by [7, IEEE802.1Q]
MII	Media Independent Interface (standardized interface provided by Ethernet controllers to access Ethernet transceivers)
MDIO	Management Data Input/Output
OA TC10	Open Alliance TC10 specification (see [8])
SVL	Shared VLAN learning defined by [7, IEEE802.1Q]

Table 2.1: Acronyms and abbreviations

2.2 Definitions

2.2.1 Ethernet packet

Definition: An "Ethernet packet" is an on-wire format defined by [9, IEEE Std 802.3-2022] which includes the following parts: Preamble (7 bytes), SFD (start frame delimiter, 1 byte), Ethernet frame (up to 2000 bytes))

2.2.2 Ethernet frame

Definition: An "Ethernet frame" is on-wire format defined [9, IEEE Std 802.3-2022] which includes the following parts: MAC destination address field (6 bytes), MAC source address field (6 bytes), Type field (2 bytes), MAC client data field (inlcude optional Q-Tag (4 bytes)) (up to 1982 bytes), optional PAD (padding bytes), FCS (frame check sequence, 4 bytes)

2.2.3 Stream

Definition: A "stream" represent multiple Ethernet frames which are grouped by similar frame attributes (e.g. MAC source address)

2.2.4 Stream identification

Definition: The term "Stream identification" is derived from [7, IEEE Std 802.1Q-2022] and represent the functionality to identify received Ethernet frames based on a particular set of frame attributes. Frames carrying different sets of frame attributes can only be identified with a single stream

3 Related documentation

3.1 Input documents & related standards and norms

[1] Layered Software Architecture
AUTOSAR_CP_EXP_LayeredSoftwareArchitecture

[2] Specification of Ethernet Interface
AUTOSAR_CP_SWS_EthernetInterface

[3] Specification of Ethernet Transceiver Driver
AUTOSAR_CP_SWS_EthernetTransceiverDriver

[4] Specification of Ethernet Driver
AUTOSAR_CP_SWS_EthernetDriver

[5] Specification of SPI Handler/Driver
AUTOSAR_CP_SWS_SPIHandlerDriver

[6] Glossary
AUTOSAR_FO_TR_Glossary

[7] IEEE 802.1Q-2022 - IEEE Standard for Local and Metropolitan Area Network - Bridges and Bridged Networks
https://ieeexplore.ieee.org/

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

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

[10] General Specification of Basic Software Modules
AUTOSAR_CP_SWS_BSWGeneral

[11] Requirements on Ethernet Support in AUTOSAR
AUTOSAR_CP_RS_Ethernet

[12] General Requirements on Basic Software Modules
AUTOSAR_CP_RS_BSWGeneral

[13] IEEE 802.1Q-2018 - IEEE Standard for Local and Metropolitan Area Network - Bridges and Bridged Networks
https://standards.ieee.org/ieee/802.1Q/6844/

[14] System Template
AUTOSAR_CP_TPS_SystemTemplate

[15] IEEE 802.1CB-2017 - IEEE Standard for Local and Metropolitan Area Network - Frame Replication and Elimination for Reliability
https://ieeexplore.ieee.org/

[16] Specification of Time Synchronization over Ethernet
AUTOSAR_CP_SWS_TimeSyncOverEthernet

[17] Specification of NVRAM Manager
AUTOSAR_CP_SWS_NVRAMManager

3.2 Related specification

AUTOSAR provides a General Specification on Basic Software [10, SWS_BSWGeneral] which is also valid for Ethernet Switch Driver.

Thus, the specifications [SWS_BSWGeneral] [10], SRS_Ethernet [11] shall be considered as additional and required specification for Ethernet Switch Driver.

4 Constraints and assumptions

4.1 Constraints

The following constraints have to be considered:

- The Ethernet switch driver module is only able to handle a single thread of execution. The execution must not be pre-empted by itself.
- The implementation is limited to 10Mbit/s, 100MBit/s, 1000Mbit/s, 2.5Gbit/s, 5Gbit/s and 10Gbit/s Ethernet bandwidth and to PHYs connected via (gigabit) Media Independent Interface (xMII).
- External MACPHY connected with an Ethernet switch over SPI are not supported.
- The Ethernet switch driver do only support VLAN-aware Ethernet switches
- The Ethernet switch driver support only passive stream identification. Active stream identification is not supported and therefore features like frame replication are not supported
- Stream identification is considered in the out-facing. Thus, stream identification is applied only at ingress side
- The Ethernet switch driver support only a subset of the IEEE specified forwarding process in an Ethernet switch. Table 4.1 give an overview which IEEE specfied forwarding processing steps are covered by AUTOSAR:

IEEE specified bridge execution order	Coverage by AUTOSAR
Default priority assignment ([AC]:13.1)	n/a
Placeholder MacSEC (incl. corresponding filter mechanism)	n/a
support of the EISS ([Q]:6.9.1)	n/a
Frame Type Acceptance filter ([Q]:6.9 f))	covered (see Chapter 7.1.7.2.1)
Ingress VID translation ([Q]:6.9 f))	n/a
Port-based VLAN Classification ([Q]:6.9 d)) XOR Port-and-Protocol-based VLAN classification ([Q]:6.12)	"Port-based VLAN Classification" covered (see Chapter 7.1.7.2.1)
Priority Code Point Decoding ([Q]:6.9.3)	"derivation of priority" covered (see Chapter 7.1.7.2.1)
Priority Regeneration ([Q]:6.9.4)	covered (see Chapter 7.1.7.2.2)
Outfacing Input Stream Identification Function(s) ([CB]:9.1.1.5)	covered (see Chapter 7.1.7.2.1)
Placeholder FRER functionality	n/a



△

Infacing Output Stream Identification Function(s) ([CB]:9.1.1.2)	n/a
Active topology enforcement ([Q]:8.6.1)	n/a
Ingress filtering ([Q]:8.6.2)	covered (see Chapter 7.1.7.2.4)
Frame filtering ([Q]:8.6.3)	covered (see Chapter 7.1.7.2.5)
Egress filtering ([Q]:8.6.4)	covered (see Chapter 7.1.7.2.6)
Stream filtering ([Q]:8.6.5.3) (selection process)	covered (see Chapter 7.1.7.2.7)
Maximum SDU Size Filtering ([Q]:8.6.5.3.1)	covered (see Chapter 7.1.7.2.7)
Stream Gating ([Q]:8.6.5.4)	covered (see Chapter 7.1.7.2.7)
Flow metering ([Q]:8.6.5.5)	covered (see Chapter 7.1.7.2.7)
ATS Eligibility Time Assignment ([Q]:8.6.5.6)	covered (see Chapter 7.1.7.2.7)
Infacing Input Stream Identification Function(s) ([CB]:9.1.1.4)	n/a
Placeholder FRER functionality	n/a
Outfacing Output Stream Identification Function(s) ([CB]:9.1.1.3)	n/a (note: only passive stream identification is supported)
Queuing frames ([Q]:8.6.6)	covered (see Chapter 7.1.7.2.8)
Shapers and Transmission selection and queuing management	covered (see Chapter 7.1.7.2.9)
Priority Code Point Encoding ([Q]:6.9.3)	covered (see Chapter 7.1.7.2.10)
Egress VID translation ([Q]:6.9 g))	n/a
support of the EISS ([Q]:6.9.2)	covered "VLAN forwarding tagged or untagged" (see Chapter 7.1.7.2.10)
Placeholder MacSEC (incl. corresponding classification mechanism)	n/a

Table 4.1: AUTOSAR coverage of IEEE specified bridge execution order

4.2 Assumptions

The following assumptions have to be considered

- Depending on the Ethernet hardware, it may become necessary that implementations deviate from API specifications in respect to the asynchronous/synchronous behavior.

4.3 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 Switch Driver module.

Modules that use the Ethernet Switch Driver module:

- Ethernet Interface (`EthIf`) calls the Ethernet Switch driver for initializing and accessing the switch device.

Modules used by the Ethernet Switch Driver module:

- Ethernet Transceiver Driver (`EthTrcv`) through the abstraction API provided by the Ethernet Interface (`EthIf`) for configuring and controlling external transceivers connected to the switch.
- Ethernet Controller Driver (`Eth`) through the abstraction API provided by the Ethernet Interface (`EthIf`) for communicating with the switch if it is managed through MDIO.
- Spi Driver (`Spi`) for communicating with the switch if the switch is managed through an SPI interface.

Dependencies to other Modules:

- On certain systems the Ethernet switch might share resources with other components, and may depend on their configuration. If those resources are within the scope of other modules (e.g. PLL configuration, memory mapping, etc.) the Ethernet Switch Driver module does not take care of configuring those components but requires their preceding initialization.

6 Requirements Tracing

The following tables reference the requirements specified in [11] as well as [12] 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_MACsec_-00004]	Configure which Ethernet ports use MACsec	[SWS_EthSwt_00564] [SWS_EthSwt_00565] [SWS_EthSwt_00566] [SWS_EthSwt_00567] [SWS_EthSwt_00568] [SWS_EthSwt_00569] [SWS_EthSwt_00570] [SWS_EthSwt_00571] [SWS_EthSwt_00572] [SWS_EthSwt_00573] [SWS_EthSwt_00574] [SWS_EthSwt_00575] [SWS_EthSwt_00576] [SWS_EthSwt_00577] [SWS_EthSwt_00578] [SWS_EthSwt_00579] [SWS_EthSwt_00580] [SWS_EthSwt_00581] [SWS_EthSwt_00582] [SWS_EthSwt_00583] [SWS_EthSwt_00584] [SWS_EthSwt_00585] [SWS_EthSwt_00586] [SWS_EthSwt_00587] [SWS_EthSwt_00588] [SWS_EthSwt_00589] [SWS_EthSwt_00590] [SWS_EthSwt_00591] [SWS_EthSwt_00592] [SWS_EthSwt_00593] [SWS_EthSwt_00594] [SWS_EthSwt_00595] [SWS_EthSwt_00596] [SWS_EthSwt_00597] [SWS_EthSwt_00598] [SWS_EthSwt_00599] [SWS_EthSwt_00600]
[SRS_BSW_00003]	All software modules shall provide version and identification information	[SWS_EthSwt_00131]
[SRS_BSW_00101]	The Basic Software Module shall be able to initialize variables and hardware in a separate initialization function	[SWS_EthSwt_00006] [SWS_EthSwt_00007] [SWS_EthSwt_00008] [SWS_EthSwt_00011]
[SRS_BSW_00161]	The AUTOSAR Basic Software shall provide a microcontroller abstraction layer which provides a standardized interface to higher software layers	[SWS_EthSwt_00099] [SWS_EthSwt_00130]
[SRS_BSW_00162]	The AUTOSAR Basic Software shall provide a hardware abstraction layer	[SWS_EthSwt_00099] [SWS_EthSwt_00130]
[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_EthSwt_00022] [SWS_EthSwt_00029] [SWS_EthSwt_00035] [SWS_EthSwt_00042] [SWS_EthSwt_00049] [SWS_EthSwt_00056] [SWS_EthSwt_00090] [SWS_EthSwt_00095] [SWS_EthSwt_00129] [SWS_EthSwt_00177] [SWS_EthSwt_00186] [SWS_EthSwt_00191] [SWS_EthSwt_00202] [SWS_EthSwt_00210] [SWS_EthSwt_00215] [SWS_EthSwt_00220] [SWS_EthSwt_00225] [SWS_EthSwt_00229] [SWS_EthSwt_00230] [SWS_EthSwt_00240] [SWS_EthSwt_00243] [SWS_EthSwt_00249] [SWS_EthSwt_00253] [SWS_EthSwt_00257] [SWS_EthSwt_00261] [SWS_EthSwt_00264] [SWS_EthSwt_00268] [SWS_EthSwt_00273] [SWS_EthSwt_00287] [SWS_EthSwt_00291] [SWS_EthSwt_00297] [SWS_EthSwt_00303] [SWS_EthSwt_00308] [SWS_EthSwt_00312] [SWS_EthSwt_00317] [SWS_EthSwt_00322] [SWS_EthSwt_00327] [SWS_EthSwt_00332] [SWS_EthSwt_00338] [SWS_EthSwt_00344] [SWS_EthSwt_00350] [SWS_EthSwt_00362] [SWS_EthSwt_00370] [SWS_EthSwt_00379]

△

Requirement	Description	Satisfied by
		△ [SWS_EthSwt_00403] [SWS_EthSwt_00405] [SWS_EthSwt_00427] [SWS_EthSwt_00432] [SWS_EthSwt_00441] [SWS_EthSwt_00443] [SWS_EthSwt_00519]
[SRS_BSW_00323]	All AUTOSAR Basic Software Modules shall check passed API parameters for validity	[SWS_EthSwt_00009] [SWS_EthSwt_00154] [SWS_EthSwt_00156] [SWS_EthSwt_00157] [SWS_EthSwt_00180]
[SRS_BSW_00347]	A Naming seperation of different instances of BSW drivers shall be in place	[SWS_EthSwt_00131]
[SRS_BSW_00350]	All AUTOSAR Basic Software Modules shall allow the enabling/ disabling of detection and reporting of development errors.	[SWS_EthSwt_00386] [SWS_EthSwt_00387] [SWS_EthSwt_00389] [SWS_EthSwt_00390] [SWS_EthSwt_00391] [SWS_EthSwt_00392] [SWS_EthSwt_00393]
[SRS_BSW_00369]	All AUTOSAR Basic Software Modules shall not return specific development error codes via the API	[SWS_EthSwt_00009] [SWS_EthSwt_00128] [SWS_EthSwt_00154] [SWS_EthSwt_00156] [SWS_EthSwt_00157] [SWS_EthSwt_00164] [SWS_EthSwt_00180]
[SRS_BSW_00375]	Basic Software Modules shall report wake-up reasons	[SWS_EthSwt_00098]
[SRS_BSW_00385]	List possible error notifications	[SWS_EthSwt_00001] [SWS_EthSwt_00113] [SWS_EthSwt_00395]
[SRS_BSW_00386]	The BSW shall specify the configuration and conditions for detecting an error	[SWS_EthSwt_00016] [SWS_EthSwt_00164]
[SRS_BSW_00395]	The Basic Software Module specifications shall list all configuration parameter dependencies	[SWS_EthSwt_00165]
[SRS_BSW_00406]	API handling in uninitialized state	[SWS_EthSwt_00123]
[SRS_BSW_00413]	An index-based accessing of the instances of BSW modules shall be done	[SWS_EthSwt_00120] [SWS_EthSwt_00154] [SWS_EthSwt_00156] [SWS_EthSwt_00157] [SWS_EthSwt_00180]
[SRS_BSW_00433]	Main processing functions are only allowed to be called from task bodies provided by the BSW Scheduler	[SWS_EthSwt_00114] [SWS_EthSwt_00115]
[SRS_BSW_00472]	Avoid detection of two production errors with the same root cause.	[SWS_EthSwt_00128]
[SRS_BSW_00478]	Timing limits of main functions	[SWS_EthSwt_00115]
[SRS_Eth_00087]	Semi-Static Auto-Configuration	[SWS_EthSwt_00031] [SWS_EthSwt_00032] [SWS_EthSwt_00060] [SWS_EthSwt_00061] [SWS_EthSwt_00086] [SWS_EthSwt_00087] [SWS_EthSwt_00091] [SWS_EthSwt_00092] [SWS_EthSwt_00098] [SWS_EthSwt_00111] [SWS_EthSwt_00117] [SWS_EthSwt_00118] [SWS_EthSwt_00125] [SWS_EthSwt_00126] [SWS_EthSwt_00127] [SWS_EthSwt_00182] [SWS_EthSwt_00183] [SWS_EthSwt_00187] [SWS_EthSwt_00188] [SWS_EthSwt_00193] [SWS_EthSwt_00194] [SWS_EthSwt_00196] [SWS_EthSwt_00197] [SWS_EthSwt_00203] [SWS_EthSwt_00204] [SWS_EthSwt_00226] [SWS_EthSwt_00227] [SWS_EthSwt_00228] [SWS_EthSwt_00235] [SWS_EthSwt_00444] [SWS_EthSwt_00445] [SWS_EthSwt_00448] [SWS_EthSwt_00449] [SWS_EthSwt_00511] ▽

▽

△

Requirement	Description	Satisfied by
		△ [SWS_EthSwt_00512] [SWS_EthSwt_00513] [SWS_EthSwt_00514] [SWS_EthSwt_00515] [SWS_EthSwt_00516] [SWS_EthSwt_00517] [SWS_EthSwt_00518] [SWS_EthSwt_91051]
[SRS_Eth_00107]	The Ethernet Transceiver Driver shall support access to the wake up reason.	[SWS_EthSwt_00442] [SWS_EthSwt_91040]
[SRS_Eth_00114]	Ethernet Switch Filtering and Policing	[SWS_EthSwt_00172] [SWS_EthSwt_00173] [SWS_EthSwt_00233] [SWS_EthSwt_00491] [SWS_EthSwt_00492] [SWS_EthSwt_00493] [SWS_EthSwt_00494] [SWS_EthSwt_00601] [SWS_EthSwt_00602] [SWS_EthSwt_00604] [SWS_EthSwt_00605] [SWS_EthSwt_00606] [SWS_EthSwt_00607] [SWS_EthSwt_00608] [SWS_EthSwt_00609]
[SRS_Eth_00118]	Transparent interface to underlying EthTrcv module(s)	[SWS_EthSwt_00018] [SWS_EthSwt_00019] [SWS_EthSwt_00023] [SWS_EthSwt_00025] [SWS_EthSwt_00026] [SWS_EthSwt_00038] [SWS_EthSwt_00044] [SWS_EthSwt_00045] [SWS_EthSwt_00051] [SWS_EthSwt_00052] [SWS_EthSwt_00098] [SWS_EthSwt_00154] [SWS_EthSwt_00156] [SWS_EthSwt_00157] [SWS_EthSwt_00164] [SWS_EthSwt_00217] [SWS_EthSwt_00222] [SWS_EthSwt_00398] [SWS_EthSwt_00440] [SWS_EthSwt_91003]
[SRS_Eth_00119]	Access to hardware status of ports	[SWS_EthSwt_00037] [SWS_EthSwt_00038] [SWS_EthSwt_00098] [SWS_EthSwt_00117] [SWS_EthSwt_00118] [SWS_EthSwt_00154] [SWS_EthSwt_00203] [SWS_EthSwt_00204] [SWS_EthSwt_00430] [SWS_EthSwt_00431]
[SRS_Eth_00120]	Hardware access via MII and/or SPI	[SWS_EthSwt_00098] [SWS_EthSwt_00206] [SWS_EthSwt_00207] [SWS_EthSwt_00211] [SWS_EthSwt_00212] [SWS_EthSwt_00216] [SWS_EthSwt_00217] [SWS_EthSwt_00221] [SWS_EthSwt_00222]
[SRS_Eth_00121]	Configuration of forwarding rules	[SWS_EthSwt_00172] [SWS_EthSwt_00173] [SWS_EthSwt_00234] [SWS_EthSwt_00461] [SWS_EthSwt_00462] [SWS_EthSwt_00463] [SWS_EthSwt_00520] [SWS_EthSwt_00521] [SWS_EthSwt_00522] [SWS_EthSwt_00523] [SWS_EthSwt_00531] [SWS_EthSwt_00535] [SWS_EthSwt_00537] [SWS_EthSwt_00539] [SWS_EthSwt_00540] [SWS_EthSwt_00541] [SWS_EthSwt_00542] [SWS_EthSwt_00543] [SWS_EthSwt_00547] [SWS_EthSwt_00548] [SWS_EthSwt_00549] [SWS_EthSwt_00550] [SWS_EthSwt_00551] [SWS_EthSwt_00553] [SWS_EthSwt_00554] [SWS_EthSwt_00555] [SWS_EthSwt_00556] [SWS_EthSwt_00557] [SWS_EthSwt_00558] [SWS_EthSwt_00559] [SWS_EthSwt_00611] [SWS_EthSwt_00612] [SWS_EthSwt_00613]
[SRS_Eth_00122]	Persistent storage of configurations	[SWS_EthSwt_00086] [SWS_EthSwt_00087] [SWS_EthSwt_00091] [SWS_EthSwt_00092] [SWS_EthSwt_00098] [SWS_EthSwt_00125] [SWS_EthSwt_00126] [SWS_EthSwt_00127] [SWS_EthSwt_00182] [SWS_EthSwt_00183] [SWS_EthSwt_00192] [SWS_EthSwt_00193] [SWS_EthSwt_00194] [SWS_EthSwt_00196]

▽

△

Requirement	Description	Satisfied by
[SRS_Eth_00123]	Testing and diagnostic of switch ports	[SWS_EthSwt_00293] [SWS_EthSwt_00299] [SWS_EthSwt_00305] [SWS_EthSwt_00309] [SWS_EthSwt_00313] [SWS_EthSwt_00318] [SWS_EthSwt_00323] [SWS_EthSwt_00328] [SWS_EthSwt_00334] [SWS_EthSwt_00340] [SWS_EthSwt_00346] [SWS_EthSwt_00416] [SWS_EthSwt_00417] [SWS_EthSwt_00418] [SWS_EthSwt_00419] [SWS_EthSwt_00420] [SWS_EthSwt_00421] [SWS_EthSwt_00422] [SWS_EthSwt_00424] [SWS_EthSwt_00425] [SWS_EthSwt_00426] [SWS_EthSwt_91014] [SWS_EthSwt_91015] [SWS_EthSwt_91016] [SWS_EthSwt_91017] [SWS_EthSwt_91018] [SWS_EthSwt_91019] [SWS_EthSwt_91020] [SWS_EthSwt_91021] [SWS_EthSwt_91022] [SWS_EthSwt_91023] [SWS_EthSwt_91024] [SWS_EthSwt_91025] [SWS_EthSwt_91029] [SWS_EthSwt_91030] [SWS_EthSwt_91031] [SWS_EthSwt_91032]
[SRS_Eth_00125]	The Ethernet Switch Driver shall support switch frame management	[SWS_EthSwt_00098] [SWS_EthSwt_00240] [SWS_EthSwt_00241] [SWS_EthSwt_00242] [SWS_EthSwt_00243] [SWS_EthSwt_00378] [SWS_EthSwt_91002] [SWS_EthSwt_91004] [SWS_EthSwt_91005] [SWS_EthSwt_91006] [SWS_EthSwt_91007] [SWS_EthSwt_91008] [SWS_EthSwt_91009] [SWS_EthSwt_91010] [SWS_EthSwt_91028]
[SRS_Eth_00126]	Independent reset of host ECU and switch hardware	[SWS_EthSwt_00292] [SWS_EthSwt_91012] [SWS_EthSwt_91013]
[SRS_Eth_00128]	The Ethernet Switch Driver shall provide statistic counter values per port	[SWS_EthSwt_00106] [SWS_EthSwt_00198] [SWS_EthSwt_00199] [SWS_EthSwt_00231] [SWS_EthSwt_00372] [SWS_EthSwt_00373] [SWS_EthSwt_91000] [SWS_EthSwt_91001]
[SRS_Eth_00178]	Ethernet Switch Stream Identification	[SWS_EthSwt_00465] [SWS_EthSwt_00467] [SWS_EthSwt_00469] [SWS_EthSwt_00471] [SWS_EthSwt_00472] [SWS_EthSwt_00475] [SWS_EthSwt_00476] [SWS_EthSwt_00477] [SWS_EthSwt_00478] [SWS_EthSwt_00479] [SWS_EthSwt_00480] [SWS_EthSwt_00481] [SWS_EthSwt_00482] [SWS_EthSwt_00483] [SWS_EthSwt_00484] [SWS_EthSwt_00486] [SWS_EthSwt_00487] [SWS_EthSwt_00610]
[SRS_Eth_00179]	Ethernet Switch Transmission Selection Algorithm	[SWS_EthSwt_00539] [SWS_EthSwt_00540] [SWS_EthSwt_00541] [SWS_EthSwt_00542] [SWS_EthSwt_00543] [SWS_EthSwt_00547] [SWS_EthSwt_00548] [SWS_EthSwt_00549] [SWS_EthSwt_00550] [SWS_EthSwt_00551] [SWS_EthSwt_00553] [SWS_EthSwt_00613]
[SRS_Eth_00180]	Ethernet Switch port scheduling of egress queues	[SWS_EthSwt_00539] [SWS_EthSwt_00540] [SWS_EthSwt_00541] [SWS_EthSwt_00542] [SWS_EthSwt_00543] [SWS_EthSwt_00547] [SWS_EthSwt_00548] [SWS_EthSwt_00549] [SWS_EthSwt_00550] [SWS_EthSwt_00551] [SWS_EthSwt_00553] [SWS_EthSwt_00613] [SWS_EthSwt_00614]
[SRS_Eth_00181]	Access to hardware internal configuration	[SWS_EthSwt_00060] [SWS_EthSwt_00061] [SWS_EthSwt_00111] [SWS_EthSwt_00187] [SWS_EthSwt_00197] [SWS_EthSwt_00228] [SWS_EthSwt_00235]

▽

△

Requirement	Description	Satisfied by
[SRS_Eth_00183]	Support of Ethernet Switch reducing energy consumption	[SWS_EthSwt_00376] [SWS_EthSwt_00377]
[SRS_Eth_00185]	Ethernet Driver Frame Preemption	[SWS_EthSwt_00529] [SWS_EthSwt_00530]

Table 6.1: Requirements Tracing

7 Functional specification

7.1 Ethernet BSW stack

As part of the AUTOSAR Layered Software Architecture according to Figure 7.1, the Ethernet BSW modules also form a layered software stack.

Figure 7.1 depicts the basic Ethernet BSW stack. The EthIf module accesses several switches using one or more Ethernet Switch Driver modules. The role of the Ethernet transceiver driver is to configure and control the physical layer ports (PHY) integrated into or connected to a switch. Whereas, the role of the Ethernet switch driver is the configuration and control of the switch. In case the Ethernet interface wants to access a PHY, it has to use the APIs of the switch driver which forward the API call to the addressed transceiver driver using the EthIf driver abstraction.

By separating the transceiver driver from the switch driver, different hardware architectures will be supported. In HW-Variant 1, the PHYs are separate devices from different vendors. They are connected via MII and MDIO to a switch which is integrated into a microcontroller. In HW-Variant 2, the switch has integrated PHYs. In HW-Variant 3, the microcontroller can control the switch via MDIO or SPI and the switch has three external PHYs which can be controlled via MDIO. In this case, different Ethernet transceiver drivers might occur.

Please note that the functional behavior of the ingress and egress port of a switch is implemented in hardware in the switch devices (see [13]). Thus, the configuration from Chapter 10 in some parts has to be written to the switch device.

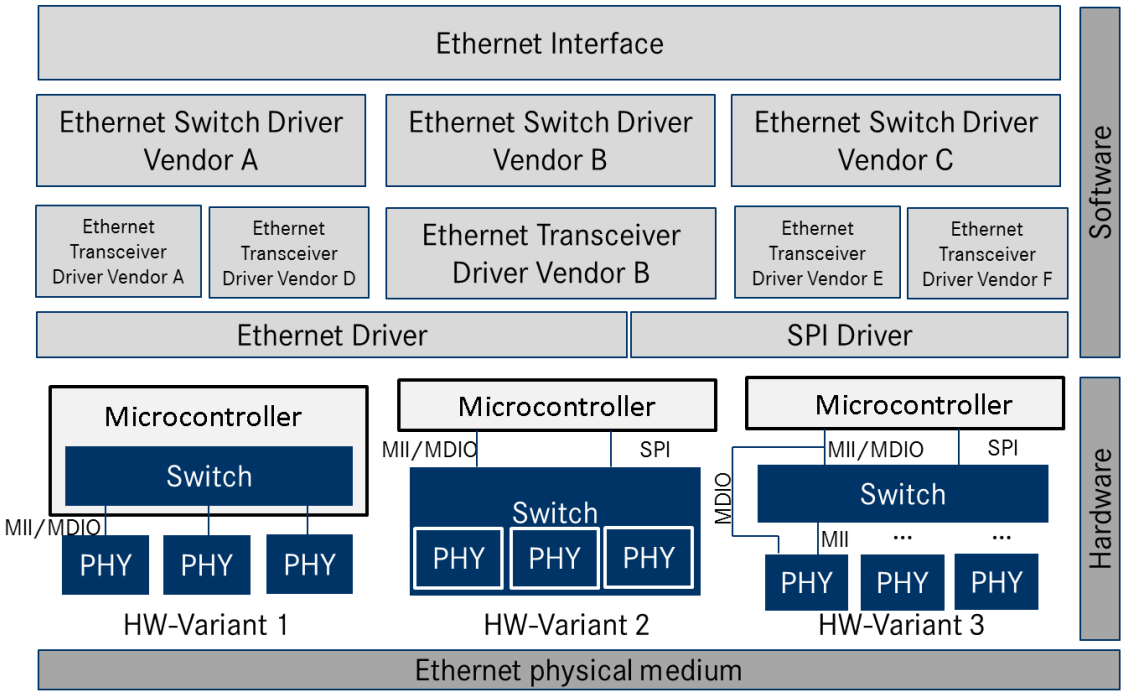

Figure 7.1: Basic Structure of the Ethernet BSW stack. (Note: The different hardware variants are alternative setups)

7.1.1 Indexing scheme

Users of the Ethernet Switch Driver identify switch resources using an indexing scheme as depicted in Figure 7.2.

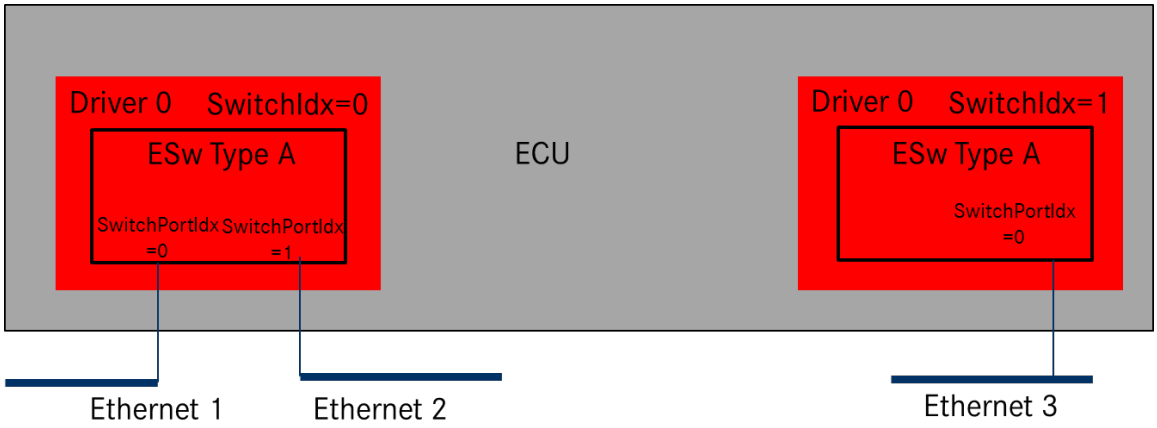

Figure 7.2: Ethernet Switch Driver indexing scheme

[SWS_EthSwt_00099]

Upstream requirements: SRS_BSW_00161, SRS_BSW_00162

⌈The Ethernet Switch Driver shall use a zero-based index to abstract the access for upper software layers.⌋

[SWS_EthSwt_00130]

Upstream requirements: SRS_BSW_00161, SRS_BSW_00162

⌈The `SwitchPortIdx` is an index for a port at the switch.⌋

[SWS_EthSwt_00120]

Upstream requirements: SRS_BSW_00413

⌈The parameter `EthSwtIdx` within the configuration shall correspond to the argument used in the API.⌋

[SWS_EthSwt_00180]

Upstream requirements: SRS_BSW_00413, SRS_BSW_00323, SRS_BSW_00369

⌈The parameter `EthSwtIndex` shall be used to distinguish different instances of a switch driver module in case the API `Det_ReportError(uint16 ModuleId, uint8 InstanceId, uint8 ApiId, uint8 ErrorId)` is called.⌋

[SWS_EthSwt_00131]

Upstream requirements: SRS_BSW_00003, SRS_BSW_00347

⌈In case different Switch devices are used in one ECU, the function names of the different Ethernet Switch drivers must be modified such that no two functions with the same names are generated. It is the responsibility of the user to take care that no two

functions with the same names are configured. The names may be extended with a vendor ID or a type ID.⌋

7.1.2 Ethernet Switch Port Mirroring

Ethernet switch port mirroring use the common established functionality of the Ethernet switch hardware to mirror traffic of one or more Ethernet switch ports (mirrored port) to a another Ethernet switch port (capture port). The mirroring configuration is given by the port mirror configuration (see [SWS_EthSwt_91017]). The port mirror configuration is set up per Ethernet switch. The configuration is stored persistently by the Ethernet switch driver. Therefore a shadow buffer is used to store the port mirror configuration during runtime and stored persistently according to the NvM storing strategy (e.g. store the shadow buffer persistently upon ECU shutdown). The port mirror configuration could be activated and de-activated, respectively, explicitly via dedicated APIs. The port mirroring is controlled by a dedicated diagnostic CDD with receive diagnostic request and forward them to the Ethernet switch driver.

[SWS_EthSwt_00416]

Upstream requirements: SRS_Eth_00123

⌈The port mirror configuration (see [SWS_EthSwt_91017]) shall be written to a shadow buffer of the Ethernet switch driver per Ethernet Switch by calling `EthSwt_WritePortMirrorConfiguration`.⌋

Note: One port mirror configuration is maintained per Ethernet switch.

[SWS_EthSwt_00417]

Upstream requirements: SRS_Eth_00123

⌈The port mirror configuration shall be enabled and disabled, respectively, per Ethernet Switch by calling `EthSwt_SetPortMirrorState`. The current state of the stored port mirror configuration shall be stored persistently, to outlast an ECU reset and to restore the port mirroring activities after an ECU reset.⌋

[SWS_EthSwt_00418]

Upstream requirements: SRS_Eth_00123

⌈The stored port mirror configuration shall be marked as "to be deleted" by calling `EthSwt_DeletePortMirrorConfiguration`, if the port mirroring of the given Ethernet switch index is disabled (see [SWS_EthSwt_91022]. Otherwise the request to delete the port mirror configuration shall be rejected.⌋

Note: The shadow buffer is stored persistently according to the NvM storing strategy, e.g. store the shadow buffer persistently upon ECU shutdown.

[SWS_EthSwt_00419]

Upstream requirements: SRS_Eth_00123

⌈The current port mirroring state shall be returned by calling `EthSwt_GetPortMirrorState`.⌋

[SWS_EthSwt_00420]

Upstream requirements: SRS_Eth_00123

⌈The port mirror configuration per Ethernet switch shall be returned by calling `EthSwt_ReadPortMirrorConfiguration`.⌋

7.1.3 State Handling

[SWS_EthSwt_00435] ⌈All functions apart from `EthSwt_SetSwitchPortMode`, `EthSwt_GetSwitchPortMode`, `EthSwt_StartSwitchPortAutoNegotiation`, `EthSwt_GetLinkState`, `EthSwt_GetBaudRate`, `EthSwt_GetDuplexMode`, `EthSwt_ReadTrcvRegister`, `EthSwt_WriteTrcvRegister`, `EthSwt_Init`, `EthSwt_MainFunction` and `EthSwt_BackgroundTask` may only be called in state `ETHSWT_STATE_ACTIVE`.

If a function which can only run (succeed with `E_OK`) in the states `ETHSWT_STATE_PORTINIT_COMPLETED` and `ETHSWT_STATE_ACTIVE` is called before state `ETHSWT_STATE_PORTINIT_COMPLETED` is reached, the Ethernet switch driver shall raise the runtime error `ETHSWT_INIT_NOT_COMPLETED`.⌋

[SWS_EthSwt_00436] ⌈`ETHSWT_STATE_PORTINIT_COMPLETED` shall be reached as soon as the port initialization has finished.⌋

Note: `ETHSWT_STATE_PORTINIT_COMPLETED` can be reached either by the function `EthSwt_Init` or by a background task (see [SWS_EthSwt_91104]).

[SWS_EthSwt_00437] ⌈`ETHSWT_STATE_ACTIVE` shall be reached, when the Ethernet switch initialization has finished.⌋

Note: The initialization of the Ethernet switch takes longer than the initialization of the Ethernet switch ports.

7.1.4 Handling of cable diagnostic

Cable diagnostic measurement is triggered by calling `EthSwt_RunPortCableDiagnostic`. The current state of the cable diagnostic measurement is polled by calling `EthSwt_GetPortCableDiagnosticsResult`. If `EthSwt_GetPortCableDiagnosticsResult` return with other value then `ETHTRCV_CABLEDIAG_PENDING`, then the cable diagnostic has finished.

Its up to the caller to re-trigger cable diagnostic again, if the measurement failed by returning ETHTRCV_CABLEDIAG_ERROR.

[SWS_EthSwt_00428] ⌈The cable diagnostic APIs (EthSwt_RunPortCableDiagnostic, EthSwt_GetPortCableDiagnosticsResult) shall only be called for Ethernet switch ports of a Ethernet switch, where the Ethernet switch ports reference an Ethernet transceiver.⌋

Note: The upper layer is a CDD that triggers the cable diagnostic measurement and maintains the cable diagnostic result. The EthSwt forwards the API calls to the EthTrcv using the EthIf driver API abstraction (see [SWS_EthSwt_00429] and [SWS_EthSwt_00346]).

7.1.5 Support of frame preemption

The EthSwt driver module supports the configuration of frame preemption per EthSwtPort (see [7, IEEE Std 802.1Q-2022]), if the Ethernet switch hardware support the functionality. By default it is assumed that Ethernet switch hardware is not supposed to perform or not even capable of frame preemption, thus EthSwtFramePreemptionEnable is set to FALSE on all EthSwtPorts.

[SWS_EthSwt_CONSTR_00527] Ethernet switch hardware support for frame preemption

Status: DRAFT

⌈If and only if an Ethernet switch hardware supports frame preemption, then it shall be allowed to set EthSwtFramePreemptionEnable to TRUE per EthSwtPort where EthSwtPortMacSecEnabled is set to FALSE.⌋

Note: Frame preemption in combination with MACsec processing is not supported in AUTOSAR.

[SWS_EthSwt_CONSTR_00528] Frame preemption enabling per EthSwtPort

Status: DRAFT

⌈If EthSwtFramePreemptionEnable is set to TRUE for an EthSwtPort, then frame preemption handling shall be enabled on ingress and on egress, for this EthSwtPort. Otherwise frame preemption handling shall be disabled for this EthSwtPort.⌋

Note: On a EthSwtPort that has frame preemption handling enabled, preemptable Ethernet frames can be preempted on egress by interspersing express Ethernet frames.

[SWS_EthSwt_00529] Frame preemption status classification of preemptable Ethernet frames on egress per traffic class

Status: DRAFT
Upstream requirements: SRS_Eth_00185

⌈If EthSwtFramePreemptionEnable is set to TRUE for an EthSwtPort, then all Ethernet frames in the EthSwtPortQueues of the corresponding EthSwtPortEgress are classified as preemptable Ethernet frames, where the EthSwtTrafficClassToPreemptionStatusAssignment is set to ETHSWT_TRAFFIC_CLASS_PREEMPTABLE.⌋

[SWS_EthSwt_00530] Frame preemption status classification of express Ethernet frames on egress per traffic class

Status: DRAFT
Upstream requirements: SRS_Eth_00185

⌈If EthSwtFramePreemptionEnable is set to TRUE for an EthSwtPort, then all Ethernet frames in the EthSwtPortQueues of the corresponding EthSwtPortEgress are classified as express Ethernet frames, where the EthSwtTrafficClassToPreemptionStatusAssignment is set to ETHSWT_TRAFFIC_CLASS_EXPRESS.⌋

Frame preemption is working on a per link basis. Thus, both ends of the link need to support the frame preemption, otherwise it does not work properly. In dynamic networks the frame preemption capability between peers is negotiated via dedicated layer 2 protocols (e.g. LLDP). However, such protocols are not supported by AUTOSAR. Automotive networks are in addition rather statically designed, rendering those protocols unnecessary, as a proper communication network design is ensured by the system description (see [14]), such that a system-wide consistent support of frame preemption can be ensured by the configuration and the data model.

7.1.6 Interaction with the Firewall module

The Ethernet frame inspection with per-stream filtering can be supported by the AUTOSAR firewall module to perform more advanced inspection techniques like stateful packet inspection and deep packet inspection. This chapter describes the functionality required by the Firewall module in the EthSwt driver to perform efficient Ethernet frame inspection and filtering supported by per-stream filtering. More details about the interaction between the firewall module and per-stream filtering can be found in CP_SWS_Firewall.

Extraction of StreamHandleIdentifier from Ethernet frame

When an Ethernet frame is passed to the firewall for inspection, it has already passed the inspection by per-stream filtering. The firewall needs to know the per-stream filtering filter rule that allows the Ethernet frame to pass. This filter rule is modeled by the StreamHandleIdentifier in the EthSwt driver. The value of the

StreamHandleIdentifier can be added by the switch core to the Ethernet frame by modifying the Ethernet frame header and adding Ethernet frame meta information to the network packet. This meta information is however not standardized and depends on the switch vendor.

[SWS_EthSwt_00500]

Status: DRAFT

⌈When EthSwt_ExtractStreamHandleIdx is called, the EthSwt driver shall extract the StreamHandleIdentifier from the passed Ethernet frame, write the value to the StreamHandleIdxPtr and return E_OK.⌋

Read out of StreamHandleIdentifier counting statistics

Many switches support counting statistics of the per-stream filtering filter rules, i.e., they count how often the filter rules provided matches to Ethernet frame. The firewall module requires this statistics information to raise Security Events (SEvs) to the IdsM. The counting statistics is typically implemented in terms of buckets, where a bucket counts the number of matches for multiple filter rules (see Figure 7.3)

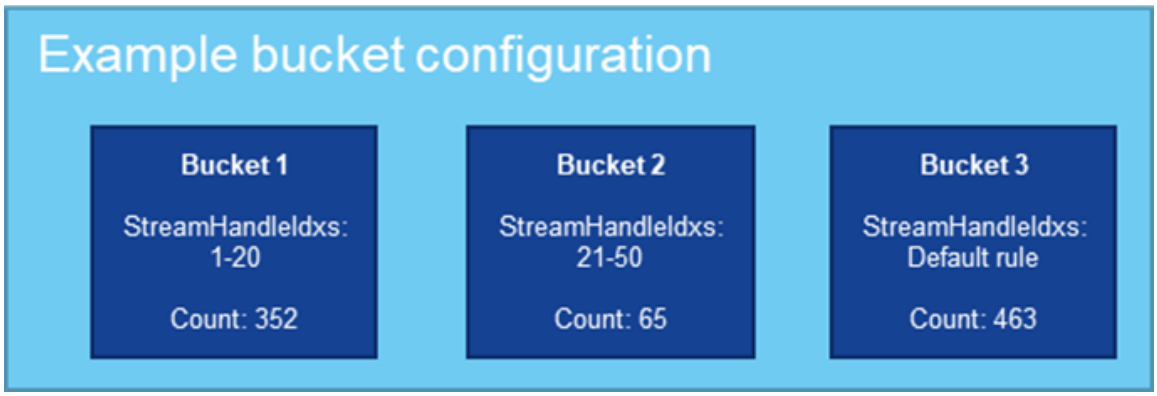

Figure 7.3: Example configuration of buckets counting matches of per-stream filtering filter rules

[SWS_EthSwt_00524] Triggering stream statistics readout

Status: DRAFT

⌈When EthSwt_GetStreamStatistics is called, the EthSwt driver shall trigger to read out the stream statistics (i.e. count values for the buckets which refer to the configured streams) from the Ethernet switch given with SwitchIdx.⌋

[SWS_EthSwt_00525] Concatenating stream statistics for buckets

Status: DRAFT

⌈When EthSwt_GetStreamStatistics has been called and the EthSwt driver performs the reading of the stream statistics, then the EthSwt driver shall concatenate the stream statistics of each available bucket as EthStreamStatisticCounter.⌋

[SWS_EthSwt_00526] Indicating the availability of stream statistics

Status: DRAFT

⌈When `EthSwt_GetStreamStatistics` has been called and the EthSwt driver finalized the reading of the stream statistics, then the EthSwt driver shall call `EthIf_StreamStatisticsIndication` with NumberOfBuckets set to the number of buckets and ListOfBucketsPtr set to the start address where the concatenated stream statistics result is stored.⌋

Depending on the switch type, the bucket values are either reset upon read-out or keeping their count values. The EthSwt driver shall provide a uniform handling of the count values independent of the switch type to ensure correct handling of the count values in the firewall module

[SWS_EthSwt_00502]

Status: DRAFT

⌈The EthSwt driver shall provide monotonically increasing count values via `EthIf_StreamStatisticsIndication` starting with 0 upon boot up. If the count values are reset by the Ethernet switch upon read-out, the EthSwt driver shall return the sum of accumulated count values since boot up and the current count values with each call of `EthIf_StreamStatisticsIndication`.⌋

Runtime management of per-stream filtering filter rules

The firewall module can be switched by the `BswM` into different states, i.e., it can be switched to enable a different set of firewall rules to be active. This use-case also extends to the per-stream filtering filter rules, which should follow the state of the firewall module to ensure coherent packet filtering.

[SWS_EthSwt_00503]

Status: DRAFT

⌈When `EthSwt_SetStreamState` is called, the EthSwt driver shall set the filter rule of the `StreamHandleIdentifier` identified by the `StreamHandleIdx` to

- Active, if `StreamActivityStatus` is set to `TRUE`
- Passive, if `StreamActivityStatus` is set to `FALSE`

⌋

[SWS_EthSwt_00504]

Status: DRAFT

⌈After successfully setting the filter rule active/passive according to [SWS_EthSwt_00503], the EthSwt driver shall call `EthIf_StreamStateIndication` for this `StreamHandleIdx` with `StreamActivityStatus` set to the current activity status of this filter rule.⌋

7.1.7 Functional Description

7.1.7.1 Learning Phase at Start-up

[SWS_EthSwt_00226]

Upstream requirements: SRS_Eth_00087

⌈The switch driver shall support a learning phase which can be divided into several sequential steps.⌋

Note: After assembly and initial power-up of the network, three learning phases follow which include MAC-Learning and IP-Address Assignment. Afterwards the learned parameters are stored to one or several non-volatile memories to make them available for subsequent start-ups. This process is shown in Figure 7.4. As an example for triggering this process, the DCM receives a diagnostic request via a bus system or a broadcast message in the Ethernet network. This diagnostic request can be forwarded to an SWC which triggers the auto-configuration process. However, the trigger is not part of this specification.

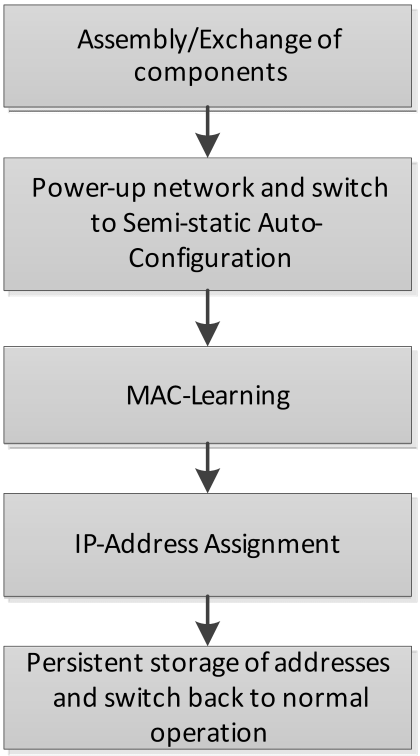

Figure 7.4: Learning Process

MAC-Learning (Optional Step): In this phase, messages need to be sent through the network and the switch will learn new MAC addresses (cf. Figure 7.5 or Figure 7.6). These MAC-addresses will be stored in addition to predefined addresses, e.g. multicast MAC addresses which are configured during the vehicle network design. If static learning is executed, i.e. MAC address will be persistently stored, it might be possible to add dynamically learned entries in the tables.

If software MAC learning is supported by switch hardware and the switch hardware expects an external microcontroller (see Variant 2 and 3 in Figure 7.1), packets with unknown MAC Source Address will be routed to this microcontroller. The MAC learning is done by integration code. It is intentionally not defined where this algorithm is located within the AUTOSAR stack as this might need a very time-optimized solution.

[7, IEEE802.1Q] define 2 different modes for MAC address learning phase:

- SVL (shared VLAN learning): the Ethernet switch considers the source MAC address from a received Ethernet frame and the ingress port from where the Etherent frame was received to create an entry for the ARL table
- IVL (independent VLAN learning): the Ethernet switch considers the source MAC address and the VLAN-ID from a received Ethernet frame and the ingress port from where the Etherent frame was received to create an entry for the ARL table

Ethernet switched network topologies where it is granted, that a MAC source address appears exlusively at the same Ethernet switch port only, would most likely use the SVL as MAC address learning mode. Otherwise IVL needs to be used as MAC address learning mode in order to have an unambiguous assignment of the MAC source address to an Ethernet switch port by considering additionally the VLAN-ID.

[SWS_EthSwt_00444] SVL MAC address learning mode

Upstream requirements: SRS_Eth_00087

⌈The Ethernet switch shall consider the MAC source address of the received Ethernet frame and the ingress port from where the Ethernet frame was received for the creation of an entry in ARL table, if all following conditions are valid:

- `EthSwtMacAddressLearningMode` is set to `SVL`
- the address learning phase has been activated (see `EthSwt_SetMacLearningMode`)

⌋

[SWS_EthSwt_00445] IVL MAC address learning mode

Upstream requirements: SRS_Eth_00087

⌈The Ethernet switch shall consider the MAC source address of the received Ethernet frame and the VLAN-ID of the received Ethernet frame and the ingress port from where the Ethernet frame was received for the creation of an entry in the ARL table, if all following conditions are valid:

- `EthSwtMacAddressLearningMode` is set to `IVL`
- the address learning phase has been activated (see `EthSwt_SetMacLearningMode`)

⌋

As stated before the EthSwt configuration could contain predefined MAC address, which are statically added to the Ethernet switch ARL table. The configuration of those predefined MAC address is constrained by the used EthSwtMacAddressLearningMode: either having VLAN membership relation if using IVL as EthSwtMacAddressLearningMode, or having no VLAN membership relation if using SVL as EthSwtMacAddressLearningMode.

[SWS_EthSwt_CONSTR_00446] SVL predefined MAC address configuration ⌈if EthSwtMacAddressLearningMode is set to SVL, then all configured EthSwtMacForwardingTable shall have no EthSwtVlanMembershipRef configured at the affected EthSwtConfig⌋

[SWS_EthSwt_CONSTR_00447] IVL predefined MAC address configuration ⌈if EthSwtMacAddressLearningMode is set to IVL, then all configured EthSwtMacForwardingTable shall have a EthSwtVlanMembershipRef configured at the affected EthSwtConfig⌋

Figure 7.5 depict an example for a MAC address learning within the Ethernet switch where EthSwtMacAddressLearningMode is set to SVL. Please note, the figure depict only a logic view on the Ethernet switch behavoiur for the MAC address learning phase and not an implementation.

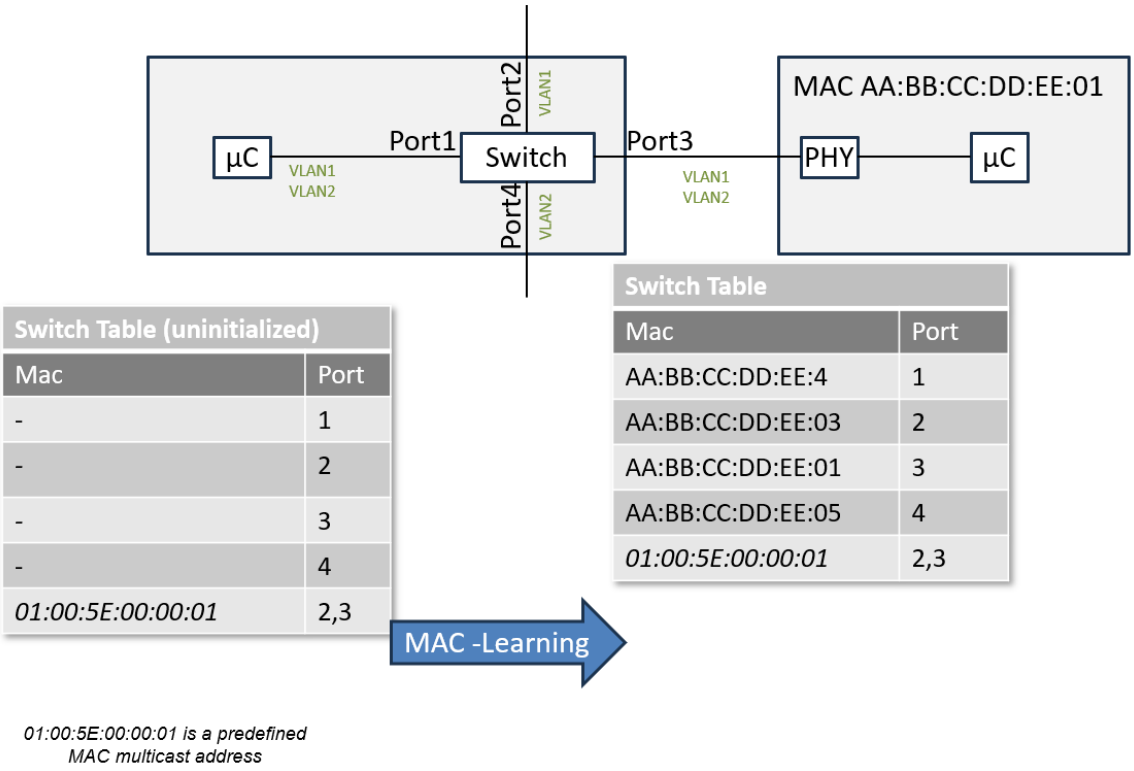

Figure 7.5: Example for a MAC address learning within the Ethernet switch with EthSwtMacAddressLearningMode is set to SVL

Figure 7.6 depict an example for a MAC address learning within the Ethernet switch where EthSwtMacAddressLearningMode is set to IVL. Please note, the figure de-

pict only a logic view on the Ethernet switch behavoiur for the MAC address learning phase and not an implementation.

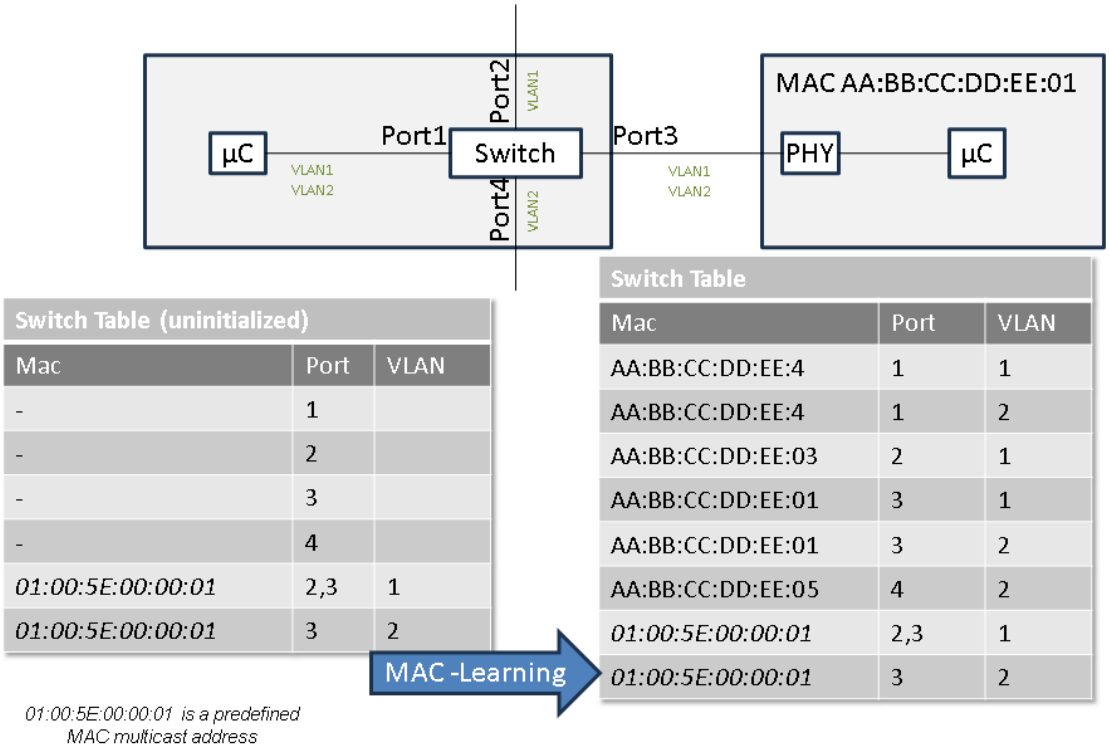

Figure 7.6: Example for a MAC address learning within the Ethernet switch with `EthSwtMacAddressLearningMode` is set to `IVL`

Note: Ingress filtering is always activated, therefore only known VLANs can be learnt

IP-Address Assignment: In this phase, ECUs without a predefined IP-address will start to acquire an IP-address via DHCP (cf. Figure 7.7). Thus, these ECUs will run a DHCP-client while the ECU with the switch will run a DHCP server. In order to be able to assign always the same IP-address to a certain node, the DHCP server needs the information at which port the MAC address has been received. This port information can be interpreted as a "domain name" in the internet which is resolved to an IP address using a domain name server (DNS). With this port information the DHCP-server will assign the IP-address according to the IP-Assignment Table to the node. As mentioned above, this allows the assignment of MAC addresses by the Tier 1 and assignment of IP addresses by the OEM. With this mechanism it is also possible to assign different IP addresses to several VLANs at the same port. For this purpose, the IP-Assignment Table needs to be extended with a VLAN-column. Please note that the MAC-Learning-Phase can be combined with this phase.

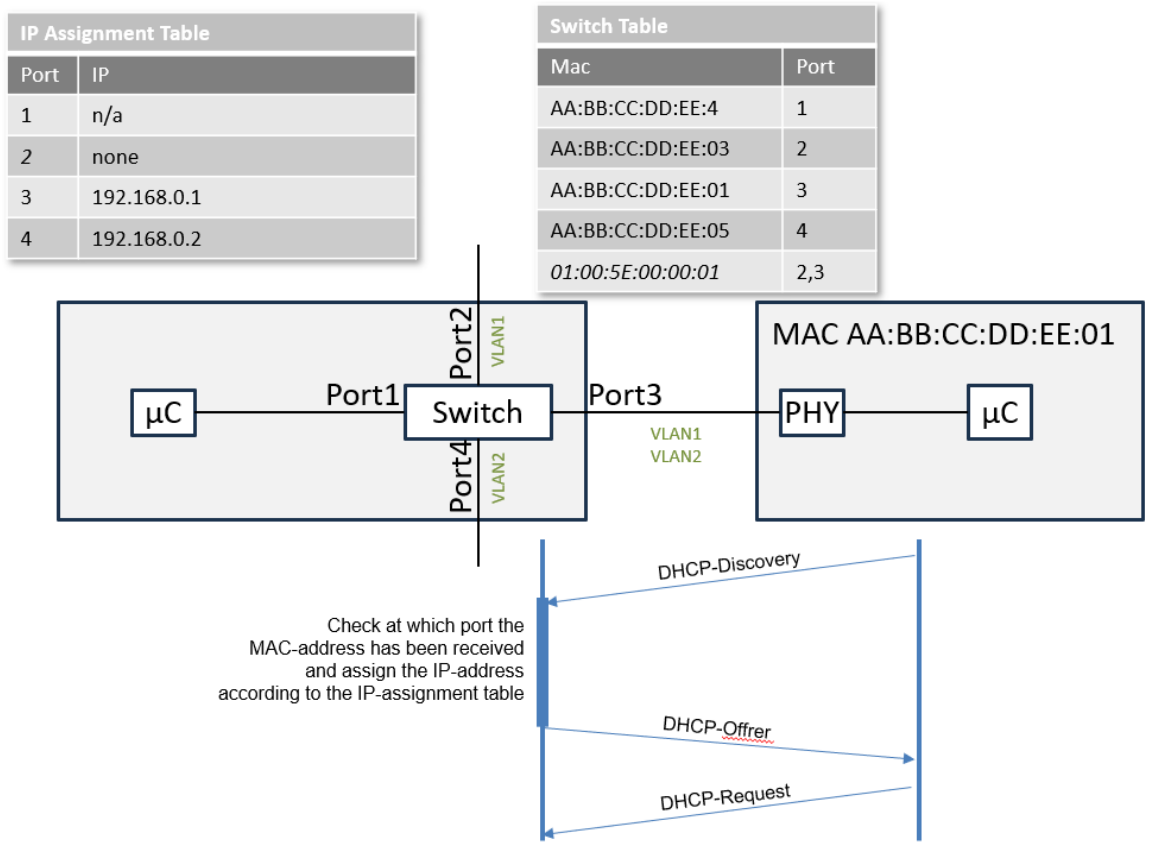

Figure 7.7: IP-address assignment via DHCP

[SWS_EthSwt_00087]

Upstream requirements: SRS_Eth_00087, SRS_Eth_00122

⌈The function `EthSwt_StoreConfiguration` shall request to store the configuration of the learned MAC/Port tables of a switch in a persistent manner. This can be done in two ways: 1.) Reading out the parameters and storing them in the NV-RAM of the host CPU using the NV-RAM manager. 2.) Advising the switch to store the configuration data in its local NV-RAM.

In both alternatives `<EthSwtPersistentConfigurationResultCallback>` shall be invoked if `EthSwtPersistentConfigurationResultCallback` is configured. In case of storage to switch local NV-RAM, `JobResult` shall be set to `NVM_REQ_OK` to indicate success or to `NVM_REQ_NOT_OK` to indicate failure.⌋

[SWS_EthSwt_00092]

Upstream requirements: SRS_Eth_00122, SRS_Eth_00087

⌈The function `EthSwt_ResetConfiguration` shall request to reset the configuration of the learned MAC/Port tables of a switch in a persistent manner. This can be done in two ways: 1.) Overwriting the learned parameters in the NV-RAM of the host CPU with preconfigured default values. 2.) Advising the switch to reset the learned configuration data in its local NV-RAM.

In both alternatives <EthSwtPersistentConfigurationResultCallback> shall be invoked if EthSwtPersistentConfigurationResultCallback is configured. In case of storage to switch local NV-RAM, JobResult shall be set to NVM_REQ_OK to indicate success or to NVM_REQ_NOT_OK to indicate failure.⌋

[SWS_EthSwt_00061]

Upstream requirements: SRS_Eth_00087, SRS_Eth_00181

⌈The function EthSwt_GetPortMacAddr shall return the port index over which the given MAC-address is reachable within the indexed switch. If for the PortIdxPtr the maximal possible value (255) is returned the given MAC address cannot be reached via a port of this switch. If multiple ports were found the API returns E_NOT_OK.⌋

[SWS_EthSwt_00163] ⌈The Ethernet Switch driver shall support an API which allows to reset learned parameters like address resolution tables by using the API EthSwt_ResetConfiguration.⌋

[SWS_EthSwt_00407] ⌈Unused ARL table entries shall be removed from the ARL table after the timeout configured via EthSwtArlTableEntryTimeout, if this parameters is present in the configuration.⌋

7.1.7.2 Frame forwarding process

As shown in Figure 7.8, the Ethernet switch consists of a certain number of Ethernet switch ports. A single physical Ethernet port is logically divided in an ingress port and an egress port. A frame is received by an Ethernet switch port in the role of an ingress port. This frame is processed within the Ethernet switch and most likely forwarded to one or more Ethernet switch ports in the role of an egress port. This process is called the "frame forwarding process". A frame forwarding process considers among others the follwoing points:

- An Ethernet frame is typically not forwarded to the Ethernet switch port where it has been received.
- A unicast Ethernet frame could be forwarded to exactly one egress port. (Please note, for some reasons (e.g. mirroring or unkown unicast Ethernet frame) a unicast Ethernet frame may forwarded to multiple egress ports)
- A multicast Ethernet frame (e.g. SOME/IP-SD offer frame) could be forwarded to one or more egress ports.
- A broadcast Ethernet frame (e.g. ARP frame) is forwarded to all egress ports except the Ethernet switch port from where the frame has been received.

Please note: The route of the frame within an Ethernet switch from an ingress port to one or multipe egress ports is called "internal frame route".

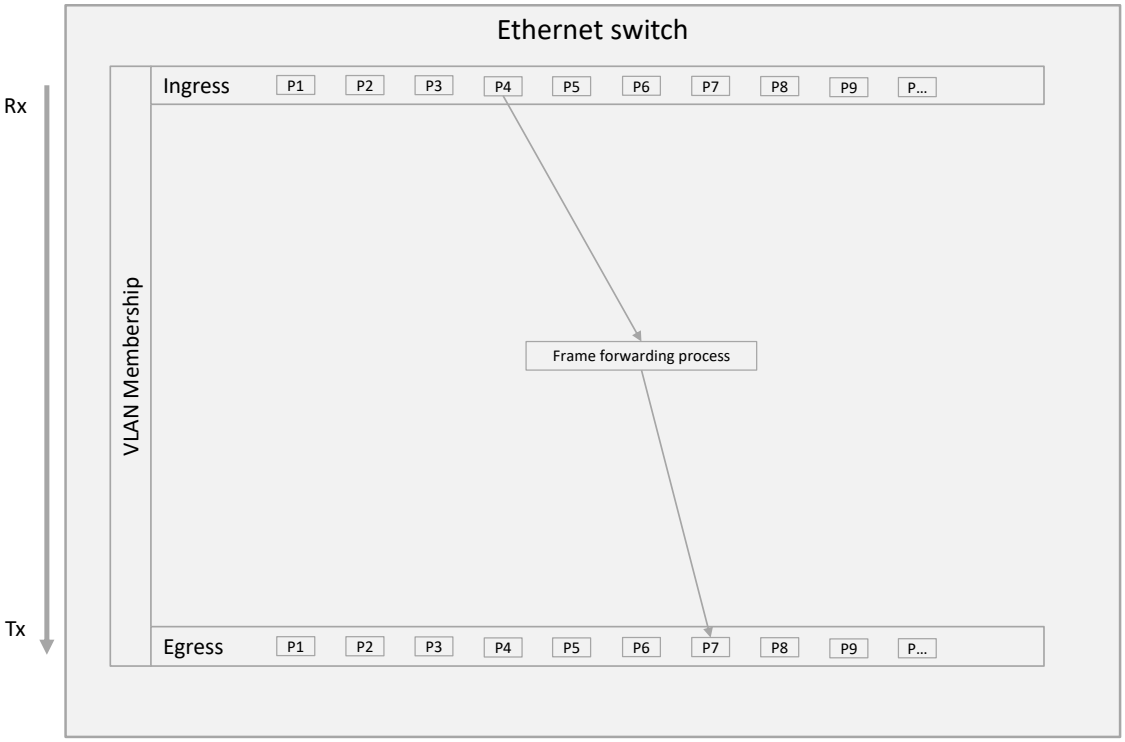

Figure 7.8: Simplified frame forwarding process within an Ethernet switch

The forwarding process consists of multiple frame processing stages. Some frame processing stages are always performed (e.g. check of VLAN membership) and some frame processing stages are performed if they are configured (e.g. flow metering). A frame processing stage may qualify a received frame as invalid. Such a frame is discarded and therefore not forwarded to the subsequential frame processing stage. [7, IEEE Std 802.1Q] specifies the frame forwarding process and particular frame processing stages. Figure 7.9 shows an overview of the processing stages which are supported by AUTOSAR (please refer to Chapter 4.1)

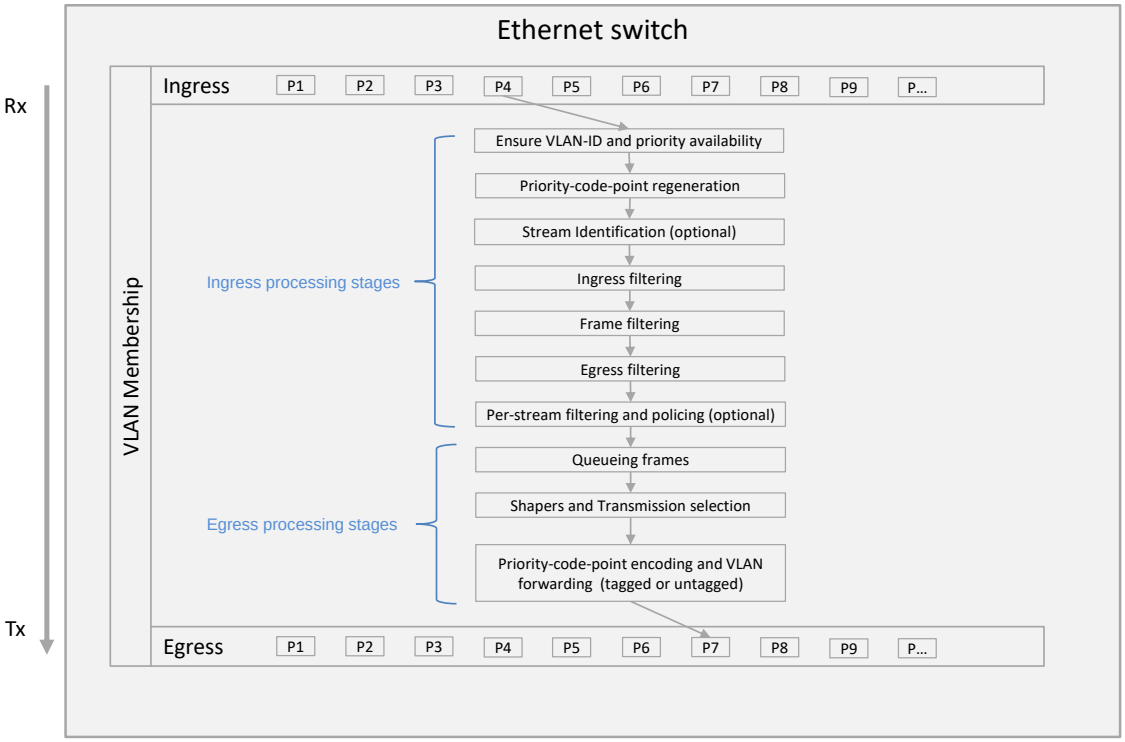

Figure 7.9: Overview of frame processing stages within an Ethernet switch supported by AUTOSAR

Most likely Ethernet frames are not modified in an Ethernet switch. Important information which impact the Ethernet frame attributes (e.g. VLAN-ID, priority) are kept in a separate memory section for each Ethernet frame while traveling through the processing stages. In this specification such a section is called "Ethernet frame meta information". If an Ethernet frame arrive at an ingress port, a Ethernet frame meta information is assigned to this Ethernet frame. The Ethernet frame meta information contain important state values (e.g. stream handle id). Available entries in the Ethernet frame meta information could be updated (e.g. destination port vector, priority). The Ethernet frame meta information is available along the internal frame route. At the very last processing stage all relevant entries of the Ethernet frame meta information which impact the Ethernet frame attributes are written to the Ethernet frame.

The following chapters describe the behaviour of the supported processing stages.

7.1.7.2.1 Ensure VLAN-ID and priority availability

AUTOSAR Ethernet switches are exclusively VLAN-aware (refer to chapter Chapter 4.1). If an Ethernet frame is received, then an AUTOSAR Ethernet switch ensures the availability of a VLAN-ID and a priority for this Ethernet frame before forwarding to the next processing stage. Therefore the so-called "frame type acceptance filter", "port-based VLAN classification" and "priority-code-point decoding" is performed by an

Ethernet switch. The following chapters describe how to ensure VLAN-ID and priority availability.

7.1.7.2.1.1 Handling of untagged Ethernet frames

Ethernet frames carrying a TPID set to 0x8100 are considered as tagged Ethernet frames, i.e. they carry a VLAN-tag. For Ethernet frames which are received without an VLAN-tag, a specific Ethernet switch handling could be configured via the AUTOSAR Ethernet switch driver.

There are two ways to handle untagged Ethernet frames at ingress side:

- Drop all untagged Ethernet frames at ingress side of the Ethernet port where the Ethernet frame was received
- Tag all untagged Ethernet frames at ingress side with a default VLAN and default VLAN priority.

Note: The handling of untagged Ethernet frames by the Ethernet switch is expected to be performed before all other modifications of the VLAN (e.g. VLAN modification). This applies also for the VLAN priority handling, which is expected to be performed before a Traffic Class assignement (see subsequential chapters) is done.

Basically, an Ethernet switch tag all Ethernet frames internally for its internal processing with hardware specific default value for a VLAN-tag. This hardware specific default value can be overwritten via `EthSwtPortIngressDefaultVlan` and `EthSwtPortIngressDefaultPriority` per Ethernet switch port

[SWS_EthSwt_CONSTR_00452]

Status: DRAFT

⌈If `EthSwtPortIngressDefaultVlan` and `EthSwtPortIngressDefaultPriority` for a particular Ethernet port is available, then the default VLAN and the default priority of this hardware Ethernet switch port shall be configured with the available values.⌋

Implementation Hint: A VLAN-tag consist of a VLAN-ID and the VLAN priority.

[SWS_EthSwt_CONSTR_00453]

Status: DRAFT

⌈A configuration of an ingress port shall be rejected as invalid, where `EthSwtPortIngressDropUntagged` is set to `FALSE` and both parameters `EthSwtPortIngressDefaultVlan` and `EthSwtPortIngressDefaultPriority` are not configured.⌋

[SWS_EthSwt_00611]

Status: DRAFT
Upstream requirements: SRS_Eth_00121

⌈If parameter EthSwtPortIngressDropUntagged of an specific ingress port is set to TRUE and a Ethernet frame without a VLAN-tag (untagged Ethernet frame) is received at this specific ingress port, then this Ethernet frame shall be dropped.⌋

[SWS_EthSwt_CONSTR_00454]

Status: DRAFT

⌈A configuration of an specific ingress port shall either have both parameters EthSwtPortIngressDefaultVlan and EthSwtPortIngressDefaultPriority configured or none of them.⌋

[SWS_EthSwt_00612]

Status: DRAFT
Upstream requirements: SRS_Eth_00121

⌈If parameter EthSwtPortIngressDropUntagged of an specific ingress port is set to FALSE and an Ethernet frame without a VLANtag (untagged Ethernet frame) is received at this specific ingress port then EthSwtPortIngressDefaultVlan and EthSwtPortIngressDefaultPriority shall be assigned to this Ethernet frame and handled for further processing by the Ethernet switch.⌋

Note: If a Ethernet frame shall be sent without a VLAN-tag (untagged Ethernet frame) of a specific VLAN-ID and on a particular egress port, then EthSwtVlanForwardingType of this VLAN-ID at this Ethernet port need to be set to ETHSWT_SENT_UNTAGGED (see Chapter 7.1.7.2.4.1)

7.1.7.2.1.2 Handling of double tagged Ethernet frames

AUTOSAR support to configure the handling for so-called "double tagged" Ethernet frames per Ethernet switch. Double tagged Ethernet frames contain two VLAN-tags. The first tag is called "S-TAG" (service provider tag) and the second tag is called "C-TAG" (customer tag). Per default the forwarding of double tagged frames is supported. In the forwarding process the S-TAG is considered. For some use cases it is necessary to avoid handdling of such Ethernet frames. Therefore a boolean parameter EthSwtDropDoubleTagged is available. The Ethernet Switch Driver supports a configuration of dropping double tagged frames via the configuration parameter EthSwtDropDoubleTagged, if the Ethernet switch hardware supports dropping of double tagged Ethernet frames.

[SWS_EthSwt_00233]

Upstream requirements: SRS_Eth_00114

⌈If parameter EthSwtDropDoubleTagged is set to TRUE, double tagged Ethernet frames shall be dropped independent on which Ethernet switch port this Ethernet frame has been received.⌋

Note: Dropping of double tagged Ethernet frames depend on the configuration of the TPID for the outer VLAN-tag.

7.1.7.2.2 Priority-Code-Point-Regeneration

If an Ethernet frame pass the processing stage to ensure VLAN-ID and priority availability, then the co-called "priority regeneration" is performed. This processing step is mandatory and will always be executed. The PCP-field (priority code point) within an VLAN-tag of an received Ethernet frame can be modified at an ingress port of an Ethernet switch. For this purpose a so-called priority regeneration table has to be defined:

Ingress PCP	0	1	2	3	4	5	6	7
Regenerated PCP	0	1	2	3	4	5	6	7

Table 7.1: Priority Regeneration: mapping the "Ingress PCP" to the "Regenerated PCP".

[SWS_EthSwt_00614]

Status: DRAFT
Upstream requirements: SRS_Eth_00180

⌈If an Ethernet frame is forwarded within an Ethernet switch, then the Ethernet switch shall perform a PCP regeneration for the PCP of this Ethernet frame by considering the configured priority regeneration table (see EthSwtPortPriorityRegeneration,EthSwtPortPriorityRegenerationIngressPCP and EthSwtPortPriorityRegenerationRegeneratedPriority) available at the EthSwtPortIngress, where this Ethernet frame was received.⌋

Please note: If no modification is required, than the PCP ingress and PCP regenerated should have the same value.

7.1.7.2.3 Stream identification

If an Ethernet frame pass the "priority-code-point regeneration" than a so-called "stream identification" could be performed by an Ethernet switch, if this processing stage is configured. Otherwise the Ethernet switch forward the Ethernet frame to the next processing stage "ingress filtering".

[15, IEEE Std 802.1CB] defines stream identification. A stream identification is the mandatory pre-condition to perform "per-stream filtering and policing" in a later pro-

cessing stage. The stream identification function is used to identify a Ethernet frame according particular Ethernet frame attributes. If the Ethernet frame match, then a so-called "stream handle id" is assigned to the Ethernet frame. Therefore the stream handle id is added to Ethernet frame meta information. If the Ethernet frame reaches the processing stage "per-stream filtering and policing", then this stream handle id is used to find a corresponding EthSwtStreamFilterEntry to perform e.g. a flow metering.

If the Ethernet switch HW supports this feature, then it can be configured by using the sub container EthSwtStreamIdentificationTable. EthSwtStreamIdentificationTable represents a table, where multipe EthSwtStreamIdentificationEntrys form an ordered list. Each entry represents an stream identification definition. The stream identification definition applies to streams within an Ethernet switch. Ethernet frames (i.e. streams) are received by an ingress port. A stream identification could be defined in dependency of ingress ports or independent of ingress ports. Therefore a stream identification could reference 0...n ingress ports (EthSwtPortIngress).

[SWS_EthSwt_CONSTR_00464]

Status: DRAFT

⌈If an EthSwtStreamIdentificationEntry is configured, then it shall be possible that this EthSwtStreamIdentificationEntry could reference none, one or multiple ingress via EthSwtStreamIdentificationIngressPortRef.⌋

[SWS_EthSwt_00465]

Status: DRAFT
Upstream requirements: SRS_Eth_00178

⌈If an EthSwtStreamIdentificationEntry references one or multiple ingress ports, then the stream identification shall be processed for streams received via any of the referencing ingress ports.⌋

[SWS_EthSwt_00467]

Status: DRAFT
Upstream requirements: SRS_Eth_00178

⌈If an EthSwtStreamIdentificationEntry references no ingress ports, then the stream identification shall be processed for all streams received via any ingress port.⌋

[SWS_EthSwt_CONSTR_00468]

Status: DRAFT

⌈If a configured EthSwtStreamIdentificationEntry references an EthSwtPortIngress, then this stream identification definition shall reference the same EthSwtPortIngress exclusively one time.⌋

The configuration of an EthSwtStreamIdentificationEntry which references ingress ports define the dependency between an EthSwtStreamIdentification-

Entry and the stream route of a received Ethernet frame (i.e. stream) within an Ethernet switch. Therefore this stream route is called the "internal stream route".

[SWS_EthSwt_00469]
Status: DRAFT
Upstream requirements: SRS_Eth_00178

⌈If an Ethernet frame (i.e stream) has been received, then the stream identification shall be processed by configured EthSwtStreamIdentificationEntry where the internal stream route match.⌋

Multiple EthSwtStreamIdentificationEntrys are configured as an ordered list of an EthSwtStreamIdentificationTable. The position within the ordered list is defined with the configured value of EthSwtStreamIdentificationPosition. The list is processed in ascending order by the Ethernet switch. As soon as the first EthSwtStreamIdentificationEntry matches, the Ethernet switch will treat this Ethernet frame according to the configuration that is associated with this EthSwtStreamIdentificationEntry. Subsequential EthSwtStreamIdentificationEntrys of the ordered list will not be applied. In case a received Ethernet frame does not match any of the EthSwtStreamIdentificationEntrys, the Ethernet frame will be forwarded to the next frame processing stage.

[SWS_EthSwt_CONSTR_00470]

Status: DRAFT

⌈Every EthSwtStreamIdentificationEntry shall have a unique position value configured via EthSwtStreamIdentificationPosition. The value shall start with 0 and continue in ascending order with no gaps for each subsequential EthSwtStreamIdentificationEntry.⌋

Note: The position value forms a ordered list of EthSwtStreamIdentificationEntrys

[SWS_EthSwt_00471]
Status: DRAFT
Upstream requirements: SRS_Eth_00178

⌈If a Ethernet frame (i.e stream) has been received, then the Ethernet switch shall check for a matching EthSwtStreamIdentificationEntry in ascending order according the EthSwtStreamIdentificationPosition, starting with EthSwtStreamIdentificationPosition configured with value 0.⌋

[SWS_EthSwt_00472]
Status: DRAFT
Upstream requirements: SRS_Eth_00178

⌈If a received Ethernet frame (i.e stream) does not match any EthSwtStreamIdentificationEntrys, the Ethernet frame shall be forwarded to the next frame processing stage without applying any further stream identification handling.⌋

An EthSwtStreamIdentificationEntry consist of the EthSwtStreamFilterRule (multiplicity 1) a EthSwtStreamHandleAssignment (multiplicity 1) and additionally of the optional element EthSwtStreamFilterAction.

The elements of an EthSwtStreamIdentificationEntry define the filter rules and filter actions. The order to perform the stream identification (apply filter rules, perform filter actions and further stream handling) is statically defined.

[SWS_EthSwt_00475]

Status: DRAFT
Upstream requirements: SRS_Eth_00178

⌈If a Ethernet frame (i.e. stream) has been received and the internal stream route match to a configured EthSwtStreamIdentificationEntry, then this stream identification shall be processed in the following order:

1. Apply the EthSwtStreamFilterRule
2. If the EthSwtStreamFilterRule identifies a match, the configured stream handle id (see EthSwtStreamHandleAssignment) shall be added to the Ethernet frame meta information and, if EthSwtStreamFilterAction is configured, consider the filter action to be performed

⌋

Note:

- It is implementation specific in which processing stage a configured EthSwtStreamFilterAction is performed. For example, if the filter action EthSwtStreamFilterActionDropFrame set to TRUE and a stream is identified, then it makes sense to immediatly drop the Ethernet frame and abort the forwarding process. But if the filter action is configured to EthSwtStreamFilterActionDestinationPortModification, then the action should be considered after egress filtering is finalized.
- If a filter rule is empty (no filter rule primitves configured (see Chapter 7.1.7.2.3.1)), then the Ethernet frame (i.e. stream) pass this filter per default. Thus, the configured stream handle id (see EthSwtStreamHandleAssignment) is added to the Ethernet frame meta information. The stream processing proceed with the "ingress filtering"

An EthSwtStreamIdentificationEntry is considered as an empty stream identification definition, where none of the optional elements are defined. Thus, incoming Ethernet frames which match the internal stream route of an empty stream identification definition, always idenfied as match of this EthSwtStreamIdentificationEntry per default.

[SWS_EthSwt_00476]
Status: DRAFT
Upstream requirements: SRS_Eth_00178

⌈A configured EthSwtStreamIdentificationEntry where no optional elements are configured, shall be considered as empty stream identification, where incoming Ethernet frames always pass.⌋

7.1.7.2.3.1 Stream identification and filter rules

An EthSwtStreamIdentificationEntry has always a filter rule configured (EthSwtStreamFilterRule). EthSwtStreamFilterRule defines which parts of a received Ethernet frame are considered for the filtering (e.g. MAC source address, IP destination address a.s.o). The EthSwtStreamFilterRule could contain multiple filter rules. One particular filter rule (e.g. EthSwtStreamFilterMACDestAddress) is called a "filter rule primitive".

[SWS_EthSwt_00477]
Status: DRAFT
Upstream requirements: SRS_Eth_00178

⌈If an EthSwtStreamFilterRule have multiple filter rule primitives configured (e.g. EthSwtStreamFilterMACSrcAddress and EthSwtStreamFilterVlanId), then the filter rule primitives shall be considered as AND-linked filter rules.⌋

Note: A received Ethernet frame (i.e. stream) matches the filter, if all configured filter rule primitives are matches. E.g. if EthSwtStreamFilterMACSrcAddress and EthSwtStreamFilterVlanId is configured, then a stream matches, if the source MAC address AND the VLAN-ID match the defined values. A stream where for example only the EthSwtStreamFilterMACSrcAddress matches is considered as NOT matching Ethernet frame.

[SWS_EthSwt_00478]
Status: DRAFT
Upstream requirements: SRS_Eth_00178

⌈If a Ethernet frame (i.e. stream) has been received, the internal stream route for this Ethernet frame matches to an configured EthSwtStreamIdentificationEntry, the EthSwtStreamFilterRule of this EthSwtStreamIdentificationEntry have filter rule primitives configured (e.g. EthSwtStreamFilterMACSrcAddress and EthSwtStreamFilterVlanId) and the Ethernet frame matches all configured filter rule primitives, then this Ethernet frame shall be qualified as matching stream.⌋

[SWS_EthSwt_00479]
Status: DRAFT
Upstream requirements: SRS_Eth_00178

⌈If an Ethernet frame (i.e. stream) has been received, the internal stream route for this Ethernet frame matches to an configured EthSwtStreamIdentificationEntry and a particular filter rule primitive in the EthSwtStreamFilterRule of this entry is not configured, then this filter rule primitive shall be considered as matching filter rule primitive.⌋

Note: Not configured filter rule primitives within an existing EthSwtStreamFilterRule are called "wildcard filter rule primitives". In order to qualify an Ethernet frame (i.e. stream) as matching stream, an Ethernet frame must match all filter rule primitives, as all filter rule primitives are AND-linked. Therefore, an empty EthSwtStreamFilterRule, i.e. without any filter rule primitives configured, will match every received Ethernet frame where the internal frame route of Ethernet frame match to configuration of the according EthSwtStreamIdentificationEntry.

7.1.7.2.3.2 Stream identification and filter action

An EthSwtStreamIdentificationEntry could define a filter action (EthSwtStreamFilterAction). The filter action describe the expected behaviour, if a matching stream has been detected. A filter action always refer to the filter rule of the same EthSwtStreamIdentificationEntry.

[SWS_EthSwt_00480]
Status: DRAFT
Upstream requirements: SRS_Eth_00178

⌈If a Ethernet frame (i.e. stream) has been qualified as matching stream and an EthSwtStreamFilterAction is configured, then this filter action shall be applied on this Ethernet frame.⌋

[SWS_EthSwt_00481]
Status: DRAFT
Upstream requirements: SRS_Eth_00178

⌈If a filter action is applied on a Ethernet frame (i.e. stream) and the corresponding EthSwtStreamFilterAction has EthSwtStreamFilterActionDropFrame set to TRUE, then this Ethernet frame shall be dropped.⌋

[SWS_EthSwt_00482]
Status: DRAFT
Upstream requirements: SRS_Eth_00178

⌈If a filter action is applied on a Ethernet frame (i.e. stream) and the corresponding EthSwtStreamFilterAction EthSwtStreamFilterActionBlockSource

set to TRUE, then this Ethernet frame and all sub sequential receptions of Ethernet frames with the same source MAC address shall be blocked.⌋

[SWS_EthSwt_00483]
Status: DRAFT
Upstream requirements: SRS_Eth_00178

⌈If a filter action is applied on a Ethernet frame (i.e. stream) and the corresponding EthSwtStreamFilterAction has an EthSwtStreamFilterActionVlanModification configured, then the VLAN-ID of this Ethernet frame shall be modified with the configured VLAN-ID given by EthSwtStreamFilterActionVlanModificationVlanId.⌋

An Ethernet switch determine the egress destination of an Ethernet frame within the forwarding process. An egress destination for an Ethernet frame could include one or multiple destination ports (egress ports). This egress destination could be modified if EthSwtStreamFilterActionDestinationPortModification is configured. The egress destination which is used for the modification is configured as reference to the according egress port(s).

[SWS_EthSwt_00484]
Status: DRAFT
Upstream requirements: SRS_Eth_00178

⌈If a filter action EthSwtStreamFilterActionDestinationPortModification is configured, then the egress destination which is used for the modification shall be determined according the configured references to egress ports via EthSwtStreamFilterActionDestinationPortModificationEgressPortRef.⌋

[SWS_EthSwt_CONSTR_00485]

Status: DRAFT

⌈If a filter action EthSwtStreamFilterActionDestinationPortModification is configured, then this EthSwtStreamFilterActionDestinationPortModification shall reference the same EthSwtPortEgress via EthSwtStreamFilterActionDestinationPortModificationEgressPortRef exclusively one time.⌋

[SWS_EthSwt_00486]
Status: DRAFT
Upstream requirements: SRS_Eth_00178

⌈If a filter action is applied on a Ethernet frame (i.e. stream), the corresponding EthSwtStreamFilterAction has an EthSwtStreamFilterActionDestinationPortModification configured and the EthSwtStreamFilterActionDestinationPortModificationType is set to ETHSWT_STREAM_EGRESS_DESTINATION_OVERWRITE, then the egress destination of this Ethernet frame shall be overwritten with the configured egress destination (see EthSwtStreamFilterActionDestinationPortModificationEgressPortRef).⌋

[SWS_EthSwt_00487]
Status: DRAFT
Upstream requirements: SRS_Eth_00178

⌈If a filter action is applied on a Ethernet frame (i.e. stream), the corresponding EthSwtStreamFilterAction has an EthSwtStreamFilterAction-DestinationPortModification configured and the EthSwtStreamFilter-ActionDestinationPortModificationType is set to ETHSWT_STREAM_EGRESS_DESTINATION_EXTEND, then the egress destination of this Ethernet frame shall be extended with the configured egress destination (see EthSwtStream-FilterActionDestinationPortModificationEgressPortRef).⌋

[SWS_EthSwt_00610]
Status: DRAFT
Upstream requirements: SRS_Eth_00178

⌈If a filter action is applied on a Ethernet frame (i.e. stream), the corresponding EthSwtStreamFilterAction has an EthSwtStreamFilterAction-DestinationPortModification configured and the EthSwtStreamFilter-ActionDestinationPortModificationType is set to ETHSWT_STREAM_EGRESS_DESTINATION_LIMIT, then the egress destination of this Ethernet frame shall be extended with the configured egress destination (see EthSwtStream-FilterActionDestinationPortModificationEgressPortRef, but limited to those referenced egress ports where this Ethernet frame is allowed to be transmitted according to the egress port state (e.g. VLAN membership, assigned MAC address)).⌋

Note: use case for [SWS_EthSwt_00610] is to limit/restrict the egress ports on which packet are allowed to egress the Ethernet switch.

7.1.7.2.4 Ingress filtering

If an Ethernet frame pass the "priority-code-point regeneration" and the optional "stream identification" frame processing stage, then a so-called ingress filtering is performed by an Ethernet switch. The following sub chapters describe the details of the processing.

7.1.7.2.4.1 Vlan-Membership

For each Ethernet switch port a VLAN membership could be defined. An Ethernet switch port could be member of 0..* VLANs. The VLAN membership impacts the frame processing. A VLAN Membership describes ingress and egress behavior in terms of filtering, tagging or untagging.

Vlan-Membership - ingress implications

If an Ethernet frame has been received, then the Ethernet switch inspect the frame regarding a VLAN-tag. If an VLAN-tag within the received Ethernet frame exist, the Ethernet switch evaluates the received VLAN identifier (VLAN-ID). If the Ethernet switch port from where the Ethernet frame has been received is member of the VLAN which is associated with the received VLAN-ID, then the frame processing will continue, otherwise the frame is discarded and no further frame processing will be performed. Thus, all supported VLAN-IDs are configured in `EthSwtVlanMembership`.

Please note: VLAN-membership ingress handling is performed in ingress processing stage "ingress filtering" as shown in Figure 7.9.

[SWS_EthSwt_00601]

Status: DRAFT
Upstream requirements: SRS_Eth_00114

⌈If an Ethernet switch port, from where the Ethernet frame has been received, is member of the VLAN according to the configuration of `EthSwtVlanMembership` and the VLAN-ID of the received Ethernet match to the configured VLAN membership of this Ethernet switch port, then the frame processing shall continue, otherwise the Ethernet frame shall be discarded and frame processing aborted.⌋

Vlan-Membership - egress implications

Please note: VLAN-membership egress handling is performed in ingress processing stage "egress filtering" as shown in Figure 7.9. It is described in this section, since this gives an overview of the VLAN-membership handling. Refer to Chapter 7.1.7.2.6 for complete egress filtering function.

If a received Ethernet frame with an particular VLAN-ID passed all processing stages, the Ethernet switch has to add the frame to an egress queue according to the internal frame route. Thereby, the VLAN membership defines with `EthSwtVlanForwardingType`, if an Ethernet frame with a particular VLAN-ID shall be sent on the affected port with a VLAN-tag (`ETHSWT_SENT_TAGGED`, or if this Ethernet frame shall be sent on the affected port without the VLAN-tag (`ETHSWT_SENT_UNTAGGED`), or if this Ethernet frame shall not be sent on the affected port (`ETHSWT_NOT_SENT`).

For each VLAN-ID a table is necessary which stores at which egress port an Ethernet frame with the corresponding VLAN-ID is sent tagged, sent untagged or not sent. For an 8-port switch, this table could look like the following example where T stands for tagging, U for untagging, N for not sent and "-" not member of this VLAN:

VLAN-ID	Port Number							
	1	2	3	4	5	6	7	8
1	T	T	-	U	-	-	-	T

▽

△

2	T	U	-	T	-	-	-	N
...								
4094								

Table 7.2: VLAN Forwarding Table

Examples of communication scenarios:

- Incoming Ethernet frames which contain a VLAN-ID of e.g. 1 can be forwarded to the ports 1, 2, 4, and 8. At ports 1, 2, and 8 these Ethernet frames will be transmitted with the VLAN-tag and at port 4 the VLAN-tag will be removed. Ethernet frames which contain a VLAN-ID e.g. 1 received on ports 3,5,6 and 7 will be discarded.
- If a broadcast message with e.g. VLAN-ID 2 will be received at port 2, it will be forwarded to port 1,4 and 8. At ports 1 and 4 these Ethernet frames will be transmitted with the VLAN-tag and on port 8 it will not be send, since the forwarding type is configured with N (ETHSWT_NOT_SENT). The other ports 3, 5, 6, and 7 are not in the same VLAN. Thus, the Ethernet frame will not be forwarded to these egress ports.
- If a broadcast message with e.g. VLAN-ID 2 will be received at port 8, it will be forwarded to port 1,2 and 4. At ports 1 and 4 these Ethernet frames will be transmitted with the VLAN-tag and on port 2 the VLAN-tag will be removed. The other ports 3, 5, 6, and 7 are not in the same VLAN. Thus, the Ethernet frame will not be forwarded to these egress ports.

The table considers only messages, which contain a VLAN-ID within the Ethernet switch.

[SWS_EthSwt_00450] ⌈If an Ethernet frame has been received and the Ethernet frame passes the Egress filtering, then the Ethernet frame shall be forwarded to the Egress port according the EthSwtVlanForwardingType configuration:

- If egress port is configured to ETHSWT_SENT_TAGGED, then the Ethernet frame shall be transmitted with a VLAN tag,
- else If egress port is configured to ETHSWT_SENT_UNTAGGED, then the Ethernet frame shall be transmitted without a VLAN tag,
- else if egress port is configured to ETHSWT_NOT_SENT, then the Ethernet frame shall be dropped

⌋

Note: VLAN-Memberships of a port are modeled with the container EthSwtVlanMembership where the associated ports are referenced via EthSwtVlanMembershipPortRef and the according EthSwtVlanForwardingType is configured.

7.1.7.2.4.2 VLAN-modification at ingress side

It is possible to define a port-based modification of the VLAN-ID or an insertion of a VLAN-ID into a received Ethernet frame. (Please note, as described in Chapter 7.1.7.2, the Ethernet frame itself will not be modified, but the change is stored in the Ethernet frame meta information to be considered in the subsequential forwarding process) This is specified with another table, e.g.:

Port Number	1	2	3	4	5	6	7	8
VLAN-ID	2	-	-	6	-	-	-	-

Table 7.3: Ingress VLAN Modification/Insertion Table

In this example, all incoming Ethernet frames at port 1 will get the VLAN-ID 2 if they already had one before. At port 4, all incoming VLAN-tagged Ethernet frames will get 6 as their VLAN-ID. At the remaining ports, no VLAN-IDs will be inserted and an existing VLAN-ID in the Ethernet frame will remain without modification.

[SWS_EthSwt_00555] Translating VLAN IDs

Upstream requirements: SRS_Eth_00121

⌈If an `EthSwtPortIngressVlanTranslationTable` is configured at an ingress port, all VLAN IDs matching a configured `EthSwtIngressVlanId` shall be mapped to the corresponding `EthSwtTranslatedVlanId`.⌋

Note: The `EthSwtTranslatedVlanId` replaces the VLAN ID of a frame internally for all stages of the forwarding process after the ingress filtering stage.

[SWS_EthSwt_CONSTR_00615] DefaultVlan must be configured if used

Status: DRAFT

⌈If an `EthSwtPortIngressVlanTranslationTable` is configured at an ingress port and this table has at least one entry where `EthSwtTranslatedVlanId` is set to 0, then `EthSwtPortIngressDefaultVlan` shall be configured for the affected `EthSwtPortIngress`.⌋

[SWS_EthSwt_00556] Wildcard for VLAN ID translation

Upstream requirements: SRS_Eth_00121

⌈If an `EthSwtPortIngressVlanTranslationTable` is configured at an ingress port, but the `EthSwtIngressVlanId` is not configured in an entry of that table, then this entry is considered a wildcard entry.⌋

[SWS_EthSwt_00557] Only one wildcard entry allowed

Upstream requirements: SRS_Eth_00121

⌈An `EthSwtPortIngressVlanTranslationTable` shall not contain more than one wildcard entry.⌋

[SWS_EthSwt_00558] Wildcard match operation

Upstream requirements: SRS_Eth_00121

⌈If a wildcard entry is configured in an EthSwtPortIngressVlanTranslationTable at an ingress port, then all VLAN IDs not matching any configured EthSwtIngressVlanId value shall be mapped to the EthSwtTranslatedVlanId of the wildcard entry.⌋

[SWS_EthSwt_00559] Translating to default VLAN ID

Upstream requirements: SRS_Eth_00121

⌈If the EthSwtTranslatedVlanId is set to 0 in any entry of that table, then the EthSwtIngressVlanId value configured in this entry shall be mapped to the EthSwtPortIngressDefaultVlan upon port-based VLAN classification.⌋

7.1.7.2.4.3 Priority handling

A VLAN-tag of an Ethernet frame consist of a VLAN-ID and the VLAN priority. The VLAN priority within a VLAN-tag is called the PCP-field (priority code point). The PCP defines the priority with which this Ethernet frame shall be handled in an Ethernet network. The PCP is a 3bit value and defines the lowest priority with 0 and highest priority with 7. The prioritisation of Ethernet traffic supports the quality of service technique on a switched Ethernet network. From the Ethernet switch perspective the priority received with an Ethernet frame could be re-defined for the internal frame processing. AUTOSAR supports the following methodes to re-define the internal priority of a received Ethernet frame for the internal frame processing:

- priority regeneration based on ingress PCP (regenerated priority (see EthSwtPortPriorityRegeneration))
- definition of an internal priority value (IPV) which could be configured if stream filtering is used (see Chapter 7.1.7.2.7)

A priority regeneration based on the PCP (VLAN-priority) of the Ethernet frame is mandatory and will always be performed. After the priority regeneration has been finalized, the regenerated priority is available in the Ethernet frame meta information. Further internal priority value (IPV) could be added to the Ethernet frame meta information in a subsequential processing stage of the Ethernet frame if stream identification is used. If an IPV is available, then this value is used for the traffic class assignment of an Ethernet frame. Otherwise the regenerated priority. However, as soon as the Ethernet frame is enqueued in a egress queue and the Ethernet frame is transmitted with a VLAN-tag, the regenerated priority value is used as PCP (VLAN-priority) of the outgoing Ethernet frame, independent of the availability and value of an IPV. The usage of a priorities differs partly:

- the regenerated priority is used for the traffic class assignment of an Ethernet frame, if an IPV is not available.

- the IPV is used for the traffic class assignment of an Ethernet frame, if an IPV is available.
- the regenerated priority is always used as PCP (VLAN-priority) for an outgoing Ethernet frame transmitted with a VLAN-tag.

Internal priority value

An internal priority value could be defined by configuring an `EthSwtPSFP` (per-stream filtering and policing) in combination with an `EthSwtStreamFilterTable`, where an `EthSwtStreamFilterEntry` references an `EthSwtStreamGateEntry` which has an `EthSwtStreamGateIPV`. As described before, the internal priority value is used to assign a traffic class to an Ethernet frame. Please refer to Chapter 7.1.7.2.7 for further description regarding the configuration.

7.1.7.2.5 Frame filtering

If an Ethernet frame passes the "ingress filtering", then a so-called "frame filtering" is performed by an Ethernet switch. In this processing stage, the Ethernet switch determines the egress ports to which the Ethernet frame is forwarded, based on the destination MAC address of the frame and the internal address resolution lookup (ARL) table. Please refer to chapter Chapter 7.1.7.1 for further information regarding the process to setup the ARL table.

[SWS_EthSwt_00461]
Status: DRAFT
Upstream requirements: SRS_Eth_00121

⌈If the destination MAC address of a received Ethernet frame is a MAC unicast/multicast address and this MAC address is available in the ARL table, then the Ethernet frame shall be considered to be forwarded to the corresponding egress port(s) according to the matching ARL table entry.⌋

[SWS_EthSwt_00520] Broadcast destination
Status: DRAFT
Upstream requirements: SRS_Eth_00121

⌈If the destination MAC address of a received Ethernet frame is the MAC broadcast address (FF:FF:FF:FF:FF:FF), then this Ethernet frame shall be considered to be forwarded to all available egress ports.⌋

Note: If the destination MAC address of a received Ethernet frame is qualified as a MAC broadcast address, then this Ethernet frame is forwarded for further processing. No check in ARL table is performed.

[SWS_EthSwt_00521] Unicast destination

Status: DRAFT
Upstream requirements: SRS_Eth_00121

⌈If the destination MAC address is a unicast address which is not available in the ARL table, then the Ethernet frame shall be considered to be forwarded to all egress ports referenced via EthSwtDestPortsForUnknownUnicastMacDestAddressRef. If all egress ports are referenced (flooding) the Ethernet frame is forwarded to all ports that are member in the respective VLAN except the incoming port to support the learning process. If the reference EthSwtDestVlanForUnknownMacDestAddressRef is defined in the context of that EthSwtUnknownMacDestAddressConfig, then only Ethernet frames addressed to that VLANs shall be considered, i.e. frame handling is VLAN-specific in this case.⌋

[SWS_EthSwt_00522] Multicast destination

Status: DRAFT
Upstream requirements: SRS_Eth_00121

⌈If the destination MAC address is a multicast address and it is not available in the ARL table, then the Ethernet frame shall be considered to be forwarded to all egress ports referenced via EthSwtDestPortsForUnknownMulticastMacDestAddressRef. If all egress ports are referenced (flooding) the Ethernet frame is forwarded to all ports that are member in the respective VLAN except the incoming port. If the reference EthSwtDestVlanForUnknownMacDestAddressRef is defined in the context of that EthSwtUnknownMacDestAddressConfig, then only Ethernet frames addressed to that VLANs shall be considered, i.e. frame handling is VLAN-specific in this case.⌋

[SWS_EthSwt_00523] Processing or dropping

Status: DRAFT
Upstream requirements: SRS_Eth_00121

⌈If after frame filtering an Ethernet Frame needs to be forwarded on at least one egress port or the meta information of this Ethernet frames contains a stream handle id (see `EthSwtStreamHandleAssignment`), the frame shall be further processed. Otherwise, the Ethernet frame shall be dropped, and the forwarding process shall be aborted.⌋

Note: An Ethernet frame which was identified by a stream identification (i.e. the meta data of the Ethernet frame contains a stream handle id), but has no egress port assigned upon frame filtering, is still subject to “pre-stream filtering and policing”. So, even Ethernet frames that are dropped upon frame filtering because of an unknown destination MAC address, e.g., can be mirrored to a specific egress port using a stream identification.

Note: If the Ethernet frame is considered to be forwarded to a least one egress port and MAC address learning is enabled, then the source MAC address is added into the ARL table.

7.1.7.2.6 Egress filtering

If an Ethernet frame pass the "frame filtering" than a so-called "egress filtering" is performed by an Ethernet switch. This processing stage has the focus on the VLAN membership. The previous processing stage "frame filtering" assigned this Ethernet to one or multiple egress ports. The egress filtering inspect the VLAN membership of the egress ports where the received Ethernet frame has been assigned to and the VLAN-ID of the received Ethernet frame. The egress filtering process keep the Ethernet frame assignment to those egress ports where the VLAN-ID of the received Ethernet frame and the VLAN membership of the egress port match. Otherwise the assignment of the Ethernet frame to a egress port is removed.

Please note: Handling of VLAN-membership is described in chapter Chapter 7.1.7.2.4.1.

[SWS_EthSwt_00462]
Status: DRAFT
Upstream requirements: SRS_Eth_00121

⌈If the VLAN membership of a egress port match to the VLAN-ID of a received Ethernet frame and this Ethernet frame has been assigned to this egress port, then the assignment of this Ethernet frame to this egress port shall be kept. Otherwise the assignment of this Ethernet frame to the affected egress port shall be removed.⌋

[SWS_EthSwt_00463]
Status: DRAFT
Upstream requirements: SRS_Eth_00121

⌈If after the egress filtering an Ethernet frame is still assigned to a least one egress port or the Ethernet frame meta information of this Ethernet frames contains a stream handle id (see `EthSwtStreamHandleAssignment`), the frame shall be further processed. Otherwise, the Ethernet frame shall be dropped, and the forwarding process shall be aborted.⌋

7.1.7.2.7 Per-stream filtering and policing

If an Ethernet frame pass the "egress filtering" than a so-called "per stream filtering and policing" could be performed by an Ethernet switch, if this processing stage is configured. Otherwise the Ethernet switch forward the Ethernet frame to the next processing stage "queueing frames".

[7, IEEE Std 802.1Q] defines per-stream filtering and policing. Per stream filtering and policing could be configured with `EthSwtPSFP`. The neighboring configuration container `EthSwtAtsInstanceTable` represent a table of so-called "aynchronous traffic shapers", which could be used by `EthSwtPSFP`. Both `EthSwtPSFP` and `EthSwtAtsInstanceTable` reside below the superordinated `EthSwtPSCM` (per stream classification and metering).

The EthSwtPSFP container include the following tables:

- EthSwtFilterMaxSduSizeTable, if configured at least one EthSwtFilterMaxSduSizeEntry exists
- EthSwtFlowMeteringTable, if configured at least one EthSwtFlowMeteringEntry exists
- EthSwtStreamFilterTable, if configured at least one EthSwtStreamFilterEntry exists
- EthSwtStreamGateTable, if configured at least one EthSwtStreamGateEntry exists

If the Ethernet switch HW supports this feature, then it can be configured by using the tables of EthSwtPSFP and EthSwtAtsInstanceTable.

The EthSwtStreamFilterTable represents the core table, because an entry of the EthSwtStreamFilterTable could reference one entry from the EthSwtFilterMaxSduSizeTable, EthSwtFlowMeteringTable and EthSwtStreamGateTable.

The EthSwtStreamFilterTable could have multiple EthSwtStreamFilterEntrys, where each entry represents a stream filter. EthSwtStreamFilterEntrys are configured as an ordered list. The position within the ordered list is defined with the configured value of EthSwtStreamFilterEntryPosition. The list is processed in ascending order by the Ethernet switch. As soon as the first EthSwtStreamFilterEntry matches, the Ethernet switch will treat this Ethernet frame according to the configuration that is associated with this EthSwtStreamFilterEntry. Subsequential EthSwtStreamFilterEntrys of the ordered list will not be applied. In case a received Ethernet frame does not match any of the EthSwtStreamFilterEntrys, the Ethernet frame will be forwarded to the next frame processing stage.

[SWS_EthSwt_CONSTR_00602]

Status: DRAFT

⌈Every EthSwtStreamFilterEntry shall have a unique position value configured via EthSwtStreamFilterEntryPosition. The value shall start with 0 and continue in ascending order with no gaps for each subsequential EthSwtStreamFilterEntry.⌋

Note: The position value forms a ordered list of EthSwtStreamFilterEntrys

[SWS_EthSwt_00602]

Status: DRAFT
Upstream requirements: SRS_Eth_00114

⌈If a Ethernet frame (i.e stream) arrives at processing stage per-stream filtering and policing, then the Ethernet switch shall check for a matching EthSwtStreamFilterEntry that is active (see [SWS_EthSwt_00503]) in ascending order according

the EthSwtStreamFilterEntryPosition, starting with EthSwtStreamFilterEntryPosition configured with value 0.⌋

[SWS_EthSwt_00604]
Status: DRAFT
Upstream requirements: SRS_Eth_00114

⌈If an arrived Ethernet frame (i.e stream) does not match any EthSwtStreamFilterEntrys or the matching EthSwtStreamFilterEntry is deactived (see [SWS_EthSwt_00503]), the Ethernet frame shall be forwarded to the next frame processing stage without applying any further stream filter handlings.⌋

An EthSwtStreamFilterEntry consist of EthSwtAssignedStreamHandle , EthSwtStreamFilterPriority, EthSwtFilterMaxSduSizeRef and additionally the optional references to the neighboring tables: EthSwtAssignedStreamHandle, EthSwtFlowMeteringEntryRef and EthSwtStreamGateEntryRef

A match of Ethernet frame to an stream filter is identified by considering EthSwtAssignedStreamHandle and EthSwtStreamFilterPriority

[SWS_EthSwt_00605]
Status: DRAFT
Upstream requirements: SRS_Eth_00114

⌈If an Ethernet frame (i.e. stream) arrives at processing stage per-stream-filtering-and-policing, and this Ethernet Frame carries an Ethernet frame meta information which contains a stream handle id (see EthSwtStreamHandleAssignment) assigned by the stream-identification processing stage, then the Ethernet switch shall scan the EthSwtStreamFilterTable (with respect to [SWS_EthSwt_00602]) for an EthSwtStreamFilterEntry where its EthSwtAssignedStreamHandle and EthSwtStreamFilterPriority match to the stream handle id and priority of the arrived Ethernet frame (see EthSwtStreamHandleAssignment and EthSwtPortPriorityRegenerationRegeneratedPriority). Ohterwise the per-stream-filtering-and-policing process shall be aborted for this Ethernet frame and the Ethernet frame shall be forwarded to the next processing stage.⌋

If an Ethernet switch detect an Ethernet frame which match to an EthSwtStreamFilterEntry, then the size of the Ethernet frame will be evaluated by considering the maximal acceptable Ethernet frame size referenced by the EthSwtStreamFilterEntry via EthSwtFilterMaxSduSizeRef.

[SWS_EthSwt_CONSTR_00603]
Status: DRAFT

⌈All EthSwtStreamFilterEntrys shall have a reference to a value of max-sdu-size via EthSwtFilterMaxSduSizeRef configured.⌋

Note: The definition of EthSwtFilterMaxSduSizeEntry includes the size of Preamble, SFD and minimum IPG (see Chapter 10.1.13)

Implementation hint: An Ethernet switch hardware does not need (and probably does not) consider the length of Preamble, SFD and minimum IPG in its native filtering mechanism. In general, these three elements can be considered as known constants in an engineered Ethernet network by the Ethernet switch engine, such that an Ethernet switch can easily be configured to behave according to the definition of EthSwtFilterMaxSduSizeEntry

[SWS_EthSwt_00606]

Status: DRAFT
Upstream requirements: SRS_Eth_00114

⌈If an Ethernet switch detect a match of an Ethernet frame to an EthSwtStreamFilterEntry according to [SWS_EthSwt_00605], then the Ethernet switch shall evaluate, if the frame size of this Ethernet frame (i.e. stream) exceeds the value of the EthSwtFilterMaxSduSizeEntry referenced by the EthSwtStreamFilterEntry via EthSwtFilterMaxSduSizeRef:

- If the frame size of the Ethernet frame exceeds the referenced max-sdu-size and then the per-stream-filtering-and-policing process shall be aborted for this Ethernet frame and the Ethernet frame shall be dropped.
- If the frame size of the Ethernet frame is equal or smaller than the referenced max-sdu-size, then the per-stream-filtering-and-policing processing shall continue.

⌋

[SWS_EthSwt_00607]

Status: DRAFT
Upstream requirements: SRS_Eth_00114

⌈If the evaluation of the Ethernet frame size result to continue with the per-stream-filtering-and-policing processing according to [SWS_EthSwt_00606] and the EthSwtStreamFilterEntry reference a EthSwtStreamGateEntry, then the Ethernet switch shall assign the configured internal priority value (see EthSwtStreamGateIPV) by updating the Ethernet frame meta information of this Ethernet frame⌋

Note: The internal priority value is used for the traffic class assignment (see 7.1.7.2.8 “Queueing frames”). The internal prioity value is modeled as a 32bit value, but the value is limited by the configured internal priority upper value of the Ethernet switch (see EthSwtUsedInternalPriorityUpperValue).

The state of a gate could be open or close. If a gate is open, then Ethernet frames could pass through for further processing. Otherwise a gate is closed and Ethernet frames are not permitted to pass through. Please note, AUTOSAR supports only open gates.

[SWS_EthSwt_CONSTR_00489]

Status: DRAFT

⌈If a EthSwtPSFP is configured, then the corresponding Ethernet switch hardware shall be configured such that Ethernet frames (i.e. streams) could always pass through (open gate).⌋

[SWS_EthSwt_00608]

Status: DRAFT
Upstream requirements: SRS_Eth_00114

⌈If the evaluation of the Ethernet frame size result to continue with the per-stream-filtering-and-policing processing according to [SWS_EthSwt_00606], then the Ethernet switch shall perform the actions in dependency of the affected EthSwtStreamFilterEntry configuration in the following order:

1. If the EthSwtStreamFilterEntry reference a EthSwtStreamGateEntry, then the Ethernet switch shall assign the conifgured internal priority value (see EthSwtStreamGateIPV) to the Ethernet frame by updating its Ethernet frame meta information.
2. If the EthSwtStreamFilterEntry reference a EthSwtFlowMeteringEntry, then the Ethernet switch shall apply the flow metering configuration on the Ethernet frame.
3. If the EthSwtStreamFilterEntry reference a EthSwtAtsInstanceEntry, then the Ethernet switch shall apply the asynchronous traffic shaper configuration on the Ethernet frame.

⌋

A EthSwtPSFP has the possibility to define a flow metering by configuring a (EthSwtFlowMeteringTable). The table contain one or multiple EthSwtFlowMeteringEntrys. Each EthSwtFlowMeteringEntry represents a configuration of one flow metering. One EthSwtStreamFilterEntry could reference excatly one EthSwtFlowMeteringEntry. The configuration of the flow metering support to limit the rate of Ethernet frames (i.e. streams).

[SWS_EthSwt_00491]

Status: DRAFT
Upstream requirements: SRS_Eth_00114

⌈If an Ethernet frame match to EthSwtStreamIdentificationEntry, this Ethernet frame pass the filtering and a EthSwtFlowMeteringEntry is available, then this Ethernet frame shall be handled by this EthSwtFlowMeteringEntry.⌋

[SWS_EthSwt_00492]
Status: DRAFT
Upstream requirements: SRS_Eth_00114

⌈A configured EthSwtFlowMeteringEntry shall perform the metering according to the configuration: EthSwtFlowMeteringColorMode, EthSwtFlowMeteringCIR, EthSwtFlowMeteringCBS, EthSwtFlowMeteringEIR, EthSwtFlowMeteringEBS and EthSwtFlowMeterCF⌋

A EthSwtPSCM has the possibility to define asynchronous traffic shaping by configuring a (EthSwtAtsInstanceTable). The table contain one or multiple EthSwtAtsInstanceEntrys. Each EthSwtAtsInstanceEntry represents a configuration of one asynchronous traffic shaper. One EthSwtStreamFilterEntry could reference excatly one EthSwtAtsInstanceEntry. The configuration of an asynchronous traffic shapping support to shape Ethernet traffic according a so-called elgibility time .

[SWS_EthSwt_00493]
Status: DRAFT
Upstream requirements: SRS_Eth_00114

⌈If asynchronous traffic shaping is configured EthSwtAtsInstanceEntry and applied on an Ethernet frame, then a elegibility time shall be assigned to this Ethernet frame by updating its Ethernet frame meta information.⌋

[SWS_EthSwt_00494]
Status: DRAFT
Upstream requirements: SRS_Eth_00114

⌈A configured EthSwtAtsInstanceEntry shall perform the scheduling according to the following configuration: EthSwtPortATSCommitedBurstSize , EthSwtPortATSCommitedInformationRate and EthSwtAtsGroupMaximumResidenceTime.⌋

Note: EthSwtAtsGroupMaximumResidenceTime is available by the referenced EthSwtAtsGroupInstanceEntry (referenced via EthSwtPortAtsSchedulerGroupRef) which is part of the EthSwtAtsGroupInstanceTable.

An EthSwtAtsGroupInstanceEntry represents one so-called "ATS Scheduler Group". All ATS instances (EthSwtAtsInstanceEntrys) which belonging to the same ATS Scheduler Group (referencing the same EthSwtAtsGroupInstanceEntry) use the same EthSwtAtsGroupMaximumResidenceTime. For an ATS Scheduler Group the eligibility assignment algorithm ensures, that Ethernet frames which have been received in a specific order will also be transmitted in that same order if they have been processed by any ATS instance belonging to that ATS scheduler group.

[SWS_EthSwt_00609]
Status: DRAFT
Upstream requirements: SRS_Eth_00114

⌈If multiple EthSwtAtsInstanceEntrys reference the same EthSwtAtsGroupInstanceEntry and Ethernet frames are processed by those EthSwtAtsInstanceEntry, then the Ethernet switch elegibility assignment algorithm shall ensure, that processed Ethernet frames are transmitted in the same order as they have been arrived at the Ethernet switch⌋

7.1.7.2.8 Queueing frames

If an Ethernet frame pass all previous processing stages of the forwarding process, then the Ethernet frames enters the egress side of the Ethernet switch and a queueing of Ethernet frames is performed. As first step a so-called "priority to traffic class assignment" is executed, i.e. the Ethernet frame will be assigned to a traffic class based on the avialable priority. The used priority for the assignment is determined from the meta information of this Ethernet frame. As stated in 7.1.7.2.4.3 “Priority handling”, the regenerated priority value (see EthSwtPortPriorityRegeneration is used if no IPV is present, otherwise the IPV value is used (added by a stream identification process (see (EthSwtStreamGateIPV), which is present in the meta information of this Ethernet frame.

[SWS_EthSwt_00531] IPV as priority
Status: DRAFT
Upstream requirements: SRS_Eth_00121

⌈If an Ethernet switch has to perform a priority to traffic class assignment for an Ethernet frame and an IPV is available in the meta information of this Ethernet frame, then the Ethernet switch shall consider the IPV value as priority. Otherwise the Ethernet switch shall consider the regenerated priority value (see EthSwtPortPriorityRegeneration) as priority.⌋

The mapping of the priority to traffic class assignment need to be configured to assign a dedicacted traffic class to an Ethernet frame based on the determined priority (see [SWS_EthSwt_00531]). The used internal priority values are configured per Ethernet switch and have a value range from 0 to EthSwtUsedInternalPriorityUpperValue. Also the used traffic classes values are configured per Ethernet switch and have a value range from 0 to EthSwtUsedTrafficClassUpperValue. The priority to traffic class mapping is configured per EthSwtPortEgress. Each EthSwtPortEgress could have multiple EthSwtPortPriorityToTrafficClassAssignments configured. One EthSwtPortPriorityToTrafficClassAssignment represents excatly one priorty to traffic class mapping, where EthSwtPortPriorityToTrafficClassAssignmentPriority represents the priority and EthSwtPortPriorityToTrafficClassAssignmentTrafficClass the traffic class. For each possible priority (range from 0

to EthSwtUsedTrafficClassUpperValue) only one EthSwtPortPriorityToTrafficClassAssignment could exist at the same EthSwtPortEgress. All Ethernet frames which arrive at an egress port are assgined to a default traffic class, where the associated priority of this Ethernet frame is not considered by an EthSwtPortPriorityToTrafficClassAssignment at this egress port. Therefore, a mandatory EthSwtPortDefaultTrafficClass is configured per EthSwtPortEgress of an Ethernet switch.

[SWS_EthSwt_CONSTR_00532] Value of priority to traffic class assignment should respect configured limitations

Status: DRAFT

⌈The value of EthSwtPortPriorityToTrafficClassAssignmentPriority within an EthSwtPortPriorityToTrafficClassAssignment shall not exceed the configured value range from 0 to EthSwtUsedInternalPriorityUpperValue at the same Ethernet switch⌋

[SWS_EthSwt_CONSTR_00533] Value of traffic class assignment should respect configured limitations

Status: DRAFT

⌈The value of EthSwtPortPriorityToTrafficClassAssignmentTrafficClass within an EthSwtPortPriorityToTrafficClassAssignment shall not exceed the configured value range from 0 to EthSwtUsedTrafficClassUpperValue at the same Ethernet switch.⌋

[SWS_EthSwt_CONSTR_00534] Availability of a default traffic class per EthSwtPortEgress

Status: DRAFT

⌈Each EthSwtPortEgress shall have a EthSwtPortDefaultTrafficClass configured. The value of EthSwtPortDefaultTrafficClass shall not exceed EthSwtUsedInternalPriorityUpperValue configured at the corresponding EthSwtConfig.⌋

For example: An Ethernet switch has configured EthSwtUsedInternalPriorityUpperValue to 10 and EthSwtUsedTrafficClassUpperValueEthSwtPortEgress to 4. Port_B of this Ethernet switch has the following EthSwtPortPriorityToTrafficClassAssignments configured:

- EthSwtPortPriorityToTrafficClassAssignment one at port_B has priority 0 to traffic class 0 assigned.
- EthSwtPortPriorityToTrafficClassAssignment two at port_B has priority 1 to traffic class 1 assigned.
- EthSwtPortPriorityToTrafficClassAssignment three at port_B has priority 2 to traffic class 1 assigned.

- EthSwtPortPriorityToTrafficClassAssignment four at port_B has priority 3 to traffic class 1 assigned.
- EthSwtPortPriorityToTrafficClassAssignment five at port_B has priority 10 to traffic class 3 assigned.
- EthSwtPortDefaultTrafficClass is configured to traffic class 2.

This priority to traffic class assignments are interpreted as depicted in Table 7.4.

Priority	Traffic Class
Prio 0	0
Prio 1-3	1
Prio 4-9	2 (default traffic class)
Prio 10	3

Table 7.4: This table shows the "Priorities" to "Traffic class" mapping according the configuration example

The previous example shows, that multiple priorities could be mapped to one traffic class. And further, only a subset of the possible priority range is configured. According the example, for Ethernet frames which are forwarded to port_B where the determined priority is in the range from 4 to 9 are assigned to the EthSwtPortDefaultTrafficClass 2, since no EthSwtPortPriorityToTrafficClassAssignment is available which cover priorities in the range from 4 to 9. Additionally, the possible traffic class 4 is not considered by the EthSwtPortPriorityToTrafficClassAssignment. This flexibility comes along with potential missconfiguration, therefore the communication network design has to ensure proper egress port configuration.

Most likely eight priorities are mapped to the eight traffic classes, since this would match to the eight VLAN-priorities of a VLAN-tag.

Table 7.5 shows an example for priority to traffic class mapping with 8 priorities, which reflect the recommended configuration described in [7, IEEE Std 802.1Q-2022] (see chapter "8.6.6. Queueing frames" Table 8-5 "Recommended priority to traffic class mappings").

Priority	Traffic Class
Prio 7	7
Prio 6	6
Prio 5	5
Prio 4	4
Prio 3	3
Prio 2	2
Prio 1	0
Prio 0	1

Table 7.5: In this table, "Priorities" are mapped to a particular "Traffic class"

[SWS_EthSwt_00535] Determine traffic class for an Ethernet frame

Status: DRAFT
Upstream requirements: SRS_Eth_00121

⌈If an Ethernet switch has determined the priority (according to [SWS_EthSwt_00531]) of an Ethernet frame and an EthSwtPortPriorityToTrafficClassAssignment is available where the EthSwtPortPriorityToTrafficClassAssignmentPriority matches to the determined priority of this Ethernet frame, then the Ethernet switch shall consider the value of the EthSwtPortPriorityToTrafficClassAssignmentTrafficClass as assigned traffic class for this Ethernet frame. Otherwise the Ethernet frame shall be assigned to the configured EthSwtPortDefaultTrafficClass.⌋

If an Ethernet frame has been succesfully assigned to an traffic class, then the Ethernet switch will try to enqueue the Ethernet frame to a EthSwtPortQueue at each EthSwtPortEgress, which is determined as destination Ethernet switch port. Each EthSwtPortQueue has one traffic class configured via EthSwtPortQueueTrafficClassAssignment.

[SWS_EthSwt_CONSTR_00536] Traffic class to egress port queue assignment

Status: DRAFT

⌈For each traffic class configured via EthSwtPortPriorityToTrafficClassAssignment and EthSwtPortDefaultTrafficClass at an EthSwtPortEgress, one EthSwtPortQueue shall be configured where the value of the EthSwtPortQueueTrafficClassAssignment is set the value of the corresponding EthSwtPortPriorityToTrafficClassAssignmentTrafficClass and EthSwtPortDefaultTrafficClass, respectively.⌋

[SWS_EthSwt_00537] Enqueue Ethernet frames to the matching egress port queue

Status: DRAFT
Upstream requirements: SRS_Eth_00121

⌈If the assigned traffic class of an Ethernet frame match to the value of the EthSwtPortQueueTrafficClassAssignment at an EthSwtPortQueue of an EthSwtPortEgress where this Ethernet frame is considered to be forwarded and the EthSwtPortQueue has free space to enqueue this transmission request, then the Ethernet switch shall enqueue this Ethernet frame in the matching EthSwtPortQueue. Otherwise the Ethernet frame shall be discarded.⌋

[SWS_EthSwt_00234]

Upstream requirements: SRS_Eth_00121

⌈The Parameter EthSwtPortQueueMinimumLength shall define the minimum length for one queue of an dedicated egress port.⌋

Note: The actual queue length can be longer. The decision on the queue length is most likely to be taken by the Ethernet switch hardware or fixed by the Ethernet switch

design. The definition of the minimum queue length in the configuration is supposed to guarantee that some priorities have enough egress buffer.

The Figure 7.10 shows the priniciple how a traffic class assignment and enqueueing of an Ethernet frame is handled in an Ethernet switch.

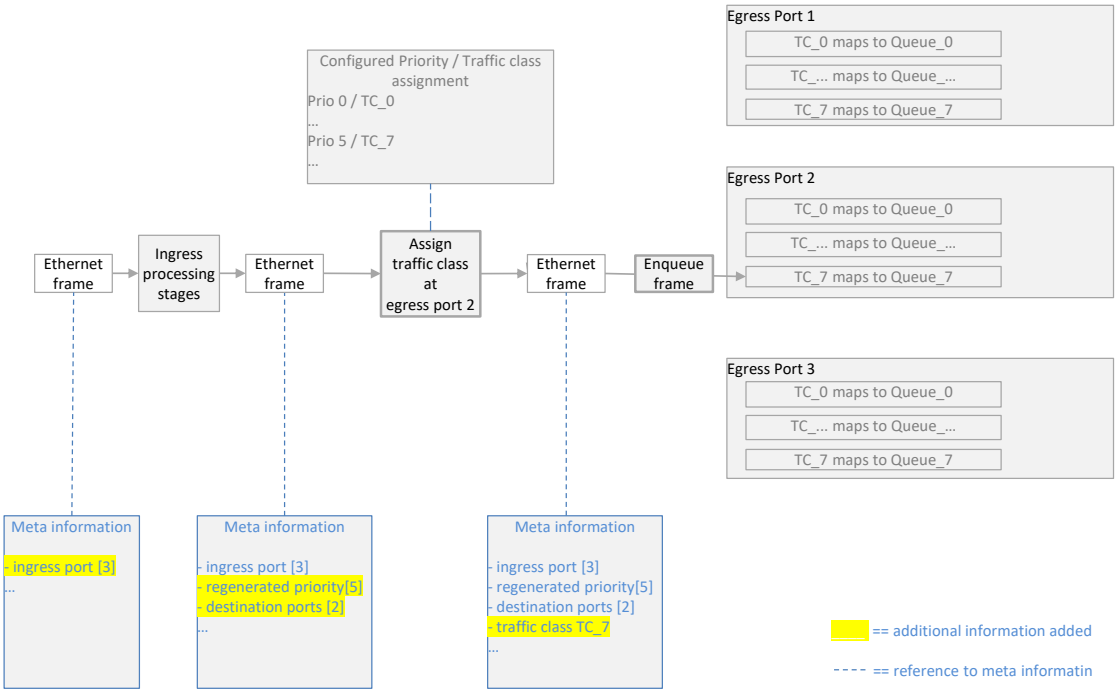

Figure 7.10: Example for an assignment to a traffic class and enqueueing of an Ethernet frame

7.1.7.2.9 Shapers and transmission selection

Ethernet frames are enqueued in egress queues according to their traffic class assignment. A Ethernet frame stay in the egress queue as long as the so-called `EthSwtPortEgressScheduler` select an Ethernet frame for transmission. Each egress queue (see `EthSwtPortQueue`) has to configure the algorithm to select the Ethernet frames for transmission. Therefore each egress queue has an mandatory sub containter `EthSwtPortEgressQueueTransmissionSelection`. `EthSwtPortEgressQueueTransmissionSelection` defines the selection algorithm via `EthSwtPortEgressQueueTransmissionSelectionAlgorithm` (e.g. credit based shaper, asynchronous traffic shaper ... a.s.o.). Each `EthSwtPortQueue` is connected to an port scheduler. The port scheduler has to schedule all connected egress queues. Each port scheduler has an mandatory sub container `EthSwtPortEgressScheduler` which defines the scheduler algorithm via `EthSwtPortSchedulerAlgorithm` (e.g. strict priority). Multiple egress schedulers at the same egress port could be configured and connected in an cascaded manner. Thus, the output of an

egress scheduler is used as an input for the sub sequential egress port scheduler. Figure 7.11 shows examples for an egress port structure.

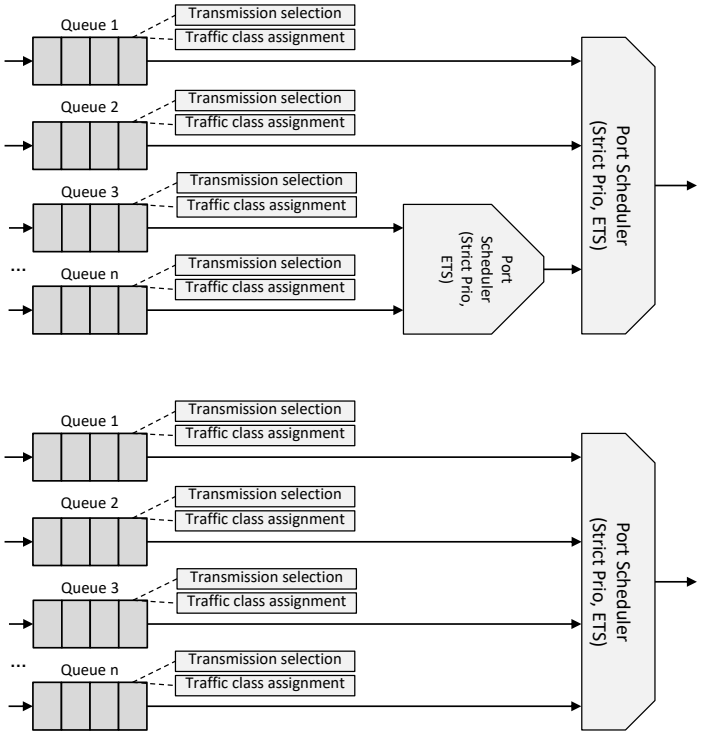

Figure 7.11: Examples for an egress port structure

The port scheduler algorithm schedule its input (either an egress queue or an egress port scheduler) by considering the according properties (e.g. traffic class assignment). Once the port scheduler algorithm has decided which of its input should be handled, the port scheduler select an Ethernet frame from the according egress queue based on the configured transmission selection algorithm:

- If the transmission selection alogrithm is configured as credit based shaper, then the according egress queue is handled as FIFO. The egress queue has an budget of credits, which is increased in the idle phase and decreased for each transmission of Ethernet frame from this egress queue.
- If the transmission selection alogrithm is configured as asynchronous traffic shaper, then the according egress queue is handled as queue. Each Ethernet frame of the queue has an assigned eligibility time. According the eligibility time a Ethernet frame is selected from this egress queue. The Ethernet frames are not handled according the arrival in this egress queue, but according the assigned eligibility time which has been added at the ingress side
- If the transmission selection alogrithm is configured as unshaped, then the according egress queue is handled as FIFO
- If the transmission selection alogrithm is configured as enhanced traffic shaping, then the according egress queue is handled is handled as queue

Note: The parameterization of the egress port influences the latency of Ethernet frames within the network.

The configuration of the egress port schedulers is done with the container EthSwtPortEgressScheduler and its sub-container EthSwtPortEgressSchedulerPredecessor with multiplicity 1 to *. Egress port scheduler connect its predecessors with the predecessor references EthSwtPortEgressPredecessorRef. An egress port scheduler could either have an further egress port scheduler or a egress port queue as predecessor.

Egress port queues are considered as neighboring egress port queues if they are referenced by the same EthSwtPortEgressScheduler via EthSwtPortEgressPredecessorRef. The composition of an egress port queue(s) and its direct connected EthSwtPortEgressScheduler form an Ethernet frame processing unit, where its output is used as input to the connected successor. The very last successor at an EthSwtPortEgress is always an EthSwtPortEgressScheduler referenced via EthSwtPortEgressLastSchedulerRef.

Please note, the configured egress port structure is an configuration model and does not reflect the hardware implementation at an egress port of an Ethernet switch.

[SWS_EthSwt_CONSTR_00538] Definition of neighboring egress port queues

Status: DRAFT

⌈Egress port queues shall be considered as neighboring egress port queues if they are referenced by the same EthSwtPortEgressScheduler via EthSwtPortEgressPredecessorRef.⌋

[SWS_EthSwt_00613]

Status: DRAFT
Upstream requirements: SRS_Eth_00121, SRS_Eth_00179, SRS_Eth_00180

⌈If an Ethernet frame is added to an EthSwtPortQueue, then the Ethernet switch shall handle this Ethernet frame according the configured transmission selection algorithm (EthSwtPortEgressQueueTransmissionSelection) of this EthSwtPortQueue and with respect to the configured egress port structure (EthSwtPortEgressScheduler, EthSwtPortEgressSchedulerPredecessor) of the corresponding egress port (EthSwtPortEgress).⌋

7.1.7.2.9.1 Details on egress port scheduler

As mentioned before EthSwtPortEgressScheduler select Ethernet frames which are offered to be transmitted by the EthSwtPortQueue based on the configured EthSwtPortEgressQueueTransmissionSelectionAlgorithm. The EthSwtPortEgressScheduler examine each of its incomming predecessor EthSwtPortQueue, starting with the EthSwtPortQueue where the highest traffic class is assigned and

proceed in descending order. This scheduling process highly depends on the configuration of the EthSwtPortSchedulerAlgorithm, the EthSwtPortEgressQueue-TransmissionSelectionAlgorithm of each relevant egress port queue and the egress port structure.

The configuration supports the follwing EthSwtPortSchedulerAlgorithms:

- ETHSWT_SCHEDULER_STRICT_PRIORITY: The egress port scheduler always selects a relevant egress port queue with the highest assigned traffic class, that offers an emission opportunity to dequeue an Ethernet frame. After each dequeued Etherent frame the scheduling algorithm checks for current available offers of an egress port queue with a higher assigned traffic class before proceeding. If no other egress port queue with a higher assigned traffic class offers an emission opportunity the scheduling alogrithm proceeds by either dequeueing further Ethernet frames of the current processed egress port queue or by scheduling the next egress port queue in descending order.
- ETHSWT_SCHEDULER_ENHANCED_TRAFFIC_SHAPER: The egress port scheduler starts with the EthSwtPortQueue where the highest traffic class is assigned and proceed in descending order. If reaching the last relevant egress port queue, the scheduling algorithm continues with the egress port queue where the highest traffic class is assigned. This round robin scheduling strictly keeps the order of the scheduled relevant egress port queues.

[SWS_EthSwt_00539] Scheduling with strict priority

Status: DRAFT
Upstream requirements: SRS_Eth_00121, SRS_Eth_00179, SRS_Eth_00180

⌈If an EthSwtPortSchedulerAlgorithm is configured with ETHSWT_SCHEDULER_STRICT_PRIORITY, then the egress port scheduler shall always select a relevant egress port queue with the highest assigned traffic class, that offers an emission opportunity to dequeue an Ethernet frame.⌋

[SWS_EthSwt_00540] Scheduling with enhandced traffic shaping

Status: DRAFT
Upstream requirements: SRS_Eth_00121, SRS_Eth_00179, SRS_Eth_00180

⌈If an EthSwtPortSchedulerAlgorithm is configured with ETHSWT_SCHEDULER_ENHANCED_TRAFFIC_SHAPER, then the egress port scheduler shall start with the egress port queue where the highest traffic class is assigned and proceed in descending order. If reaching the last relevant egress port queue, the scheduling algorithm shall continue with the egress port queue where the highest traffic class is assigned.⌋

The EthSwtPortEgressQueueTransmissionSelectionAlgorithm and the according configuration defines the amount of Etherne frames which are dequeued, if the egress port scheduler select an relevant egress port queue.

[SWS_EthSwt_00541] Dequeueing Ethernet frames with strict priority scheduling

Status: DRAFT
Upstream requirements: SRS_Eth_00121, SRS_Eth_00179, SRS_Eth_00180

⌈If an EthSwtPortSchedulerAlgorithm is configured with ETHSWT_SCHEDULER_STRICT_PRIORITY and a relevant egress port queue is selected that offers an emission opportunity, then the egress port scheduler shall dequeue Ethernet frames from this egress port queue until either of the following conditions is valid:

- an egress port queue with a higher traffic class offers an emission opportunity
- the emission offer opportunity of this egress port queue is suspended
- no further Ethernet frames are available in this egress port queue

⌋

[SWS_EthSwt_00542] Emission opportunity suspension with strict priority scheduling

Status: DRAFT
Upstream requirements: SRS_Eth_00121, SRS_Eth_00179, SRS_Eth_00180

⌈If an EthSwtPortSchedulerAlgorithm is configured with ETHSWT_SCHEDULER_STRICT_PRIORITY and the connected egress port queue is dequeued by the EthSwtPortEgressScheduler, then a connected egress port queue shall suspend its emission opportunity in dependency to the configured EthSwtPortEgressQueueTransmissionSelectionAlgorithm according the following conditions:

- If set to ETHSWT_TRANSMISSION_SELECTION_ALGORITHM_UNSHAPED, then the emission opportunity is suspended if no Ethernet frame resides in the egress port queue
- If set to ETHSWT_TRANSMISSION_SELECTION_ALGORITHM_CBS, then the emission opportunity is suspended if the credit of this egress port queue is lower than 0
- If set to ETHSWT_TRANSMISSION_SELECTION_ALGORITHM_ATS, then the emission opportunity is suspended if all Ethernet frames are dequeued which are eligible for transmission
- If set to ETHSWT_TRANSMISSION_SELECTION_ALGORITHM_ETS, then the emission opportunity is suspended under either the following conditions:
  - if the configured limit (either in weight or in percent of the available bandwidth see (EthSwtPortEgressQueueTransmissionSelectionETSConfig)) exceeds and at least one neighboring egress port queue configured to ETHSWT_TRANSMISSION_SELECTION_ALGORITHM_ETS resume its emission opportunity
  - no further Ethernet frames resides in this egress port queue

⌋

Please note: It is recommended to use ETHSWT_TRANSMISSION_SELECTION_ALGORITHM_UNSHAPED for egress port queues, where it is ensured by the network communication design, that the amount of network traffic is limited to a minium. Otherwise Ethernet frames in egress port queues assigned to lower traffic classes may be confronted with high delay in network traffic burst scenarios.

[SWS_EthSwt_00543] Emission opportunity suspension with enhanced traffic shaping

Status: DRAFT
Upstream requirements: SRS_Eth_00121, SRS_Eth_00179, SRS_Eth_00180

⌈If an EthSwtPortSchedulerAlgorithm is configured with ETHSWT_SCHEDULER_ENHANCED_TRAFFIC_SHAPER and the connected egress port queue is dequeued by the EthSwtPortEgressScheduler, then a connected egress port queue shall suspend its emission opportunity in dependency to the configured EthSwtPortEgressQueueTransmissionSelectionAlgorithm according the following conditions:

- If set to ETHSWT_TRANSMISSION_SELECTION_ALGORITHM_ETS, then the emission opportunity is suspended under either the following conditions:
  - if the configured limit (either in weight or in percent of the available bandwidth see (EthSwtPortEgressQueueTransmissionSelectionETSConfig)) exceeds and at least one neighboring egress port queue configured to ETHSWT_TRANSMISSION_SELECTION_ALGORITHM_ETS resume its emission opportunity
  - no further Ethernet frames resides in this egress port queue

⌋

An EthSwtPortEgressScheduler where the EthSwtPortSchedulerAlgorithm is configured to ETHSWT_SCHEDULER_ENHANCED_TRAFFIC_SHAPER could only handle egress port queues where the EthSwtPortEgressQueueTransmissionSelectionAlgorithm are configured to ETHSWT_TRANSMISSION_SELECTION_ALGORITHM_ETS. An egress port queue configured with ETHSWT_SCHEDULER_ENHANCED_TRAFFIC_SHAPER as its EthSwtPortSchedulerAlgorithm need to have at least one neighboring egress port queue with the same EthSwtPortSchedulerAlgorithm. An EthSwtPortEgressScheduler where the EthSwtPortSchedulerAlgorithm is configured to ETHSWT_SCHEDULER_ENHANCED_TRAFFIC_SHAPER shape the traffic of all direct connected egress port queues to get an fair bandwidth distribution in traffic contesgtion scenarios where at least two egress port queues resume the emission opportunity.

[SWS_EthSwt_CONSTR_00544] Egress configuration constraint for scheduling with enhanced traffic shaping

Status: DRAFT

⌈A configuration where an EthSwtPortEgressScheduler has set the EthSwtPortSchedulerAlgorithm to ETHSWT_SCHEDULER_ENHANCED_TRAFFIC_SHAPER shall support to have egress port queues with EthSwtPortEgressQueueTransmissionSelectionAlgorithm set to ETHSWT_TRANSMISSION_SELECTION_ALGORITHM_ETS. All other EthSwtPortEgressQueueTransmissionSelectionAlgorithms are not supported in combination with EthSwtPortEgressScheduler set to ETHSWT_SCHEDULER_ENHANCED_TRAFFIC_SHAPER.⌋

[SWS_EthSwt_CONSTR_00545] Enhanded traffic shaping require at least two egress port queues

Status: DRAFT

⌈A configuration where an EthSwtPortEgressScheduler has set the EthSwtPortSchedulerAlgorithm to ETHSWT_SCHEDULER_ENHANCED_TRAFFIC_SHAPER shall have at least two egress port queues with EthSwtPortEgressQueueTransmissionSelectionAlgorithm set to ETHSWT_TRANSMISSION_SELECTION_ALGORITHM_ETS as direct connected predecessors.⌋

The combination of EthSwtPortEgressScheduler with EthSwtPortSchedulerAlgorithm set to ETHSWT_SCHEDULER_ENHANCED_TRAFFIC_SHAPER and egress port queues with EthSwtPortEgressQueueTransmissionSelectionAlgorithm set to ETHSWT_TRANSMISSION_SELECTION_ALGORITHM_ETS realize round-robin network traffic handling. The available bandwidth for transmission of enqueued Ethernet frames of the configured egress port queues, are configured per egress port queue via EthSwtPortEgressQueueTransmissionSelectionETSConfig. The configuration supports the following options:

- configuration of available bandwidth as weights of Ethernet frames via EthSwtETSConfigAvailableBandwidthInWeightValue
- configuration of available bandwidth in percent via EthSwtETSConfigAvailableBandwidthInPercent

Independent which configuration variant for EthSwtPortEgressQueueTransmissionSelectionETSConfig is used, all egress port queues that are scheduled by the same EthSwtPortEgressScheduler with EthSwtPortEgressQueueTransmissionSelectionAlgorithm set to ETHSWT_TRANSMISSION_SELECTION_ALGORITHM_ETS should use the same variant of EthSwtPortEgressQueueTransmissionSelectionETSConfig

[SWS_EthSwt_CONSTR_00546] Neighboring egress port queues need the same variant of availability bandwidth configuration

Status: DRAFT

⌈If an EthSwtPortEgressScheduler is configured with EthSwtPortScheduler-Algorithm set to ETHSWT_SCHEDULER_ENHANCED_TRAFFIC_SHAPER, then all egress port queues which are configured for this EthSwtPortEgressScheduler shall exclusively use the same confiugration of EthSwtPortEgressQueue-TransmissionSelectionETSConfig:

- EthSwtETSConfigAvailableBandwidthInWeightValue XOR
- EthSwtETSConfigAvailableBandwidthInPercent

⌋

The configuration for the ETS traffic shaping allows the following variants:

- EthSwtETSConfigAvailableBandwidthInWeightValue: the available bandwidth is configured in weights, where the weights represents the amount of Ethernet frames
- EthSwtETSConfigAvailableBandwidthInPercent: the available bandwidth per egress port queue is configured in percent

Both configuration variants are based on congestion scenario where all neighboring egress port queues consume their bandwidth.

The configuration of the available bandwidth in weights as Ethernet frames need to be considered in relation to the emission of all neighboring egress port queues. The sum of all configured weights as Ethernet frames across all neighboring egress port queues reflect one emission portion of Ethernet frames. If an emission portion of Ethernet frames were processed by an EthSwtPortEgressScheduler, then the amount of configured weights as Ethernet frames per neighboring egress port queue should be enclosed in the emission portion of processed Ethernet frames. Or in other words, the configured available bandwidth in weights as Ethernet frames of each neighboring egress port queue should be processed, if an emission portion of Ethernet frames were processed by the EthSwtPortEgressScheduler.

The configuration of the available bandwidth in percent need to be considered in relation to a measurement interval. This interval defines the time slot which is used to calculated the expected emission of each egress port at the same EthSwtPort-EgressScheduler.

Note: For both configuration variants count, the order of Ethernet frames, either within one emission portion or within the measurement interval, depends on the implemented scheduler algorithm (e.g. weighted round robin, deficit round robin) and is not defined / configurable by the EthSwt driver module.

[SWS_EthSwt_00547] Determination of egress port queue emission with available bandwidth configured in weights as amount of Ethernet frames

Status: DRAFT
Upstream requirements: SRS_Eth_00121, SRS_Eth_00179, SRS_Eth_00180

⌈If the available bandwidth of neighboring egress port queues configured with `EthSwtETSConfigAvailableBandwidthInWeightValue` and all egress port queues consume their available bandwidth, then the expected emission of all neighboring egress port queues shall be determined with the following considerations:

The emission of one egress queue in Ethernet frames:

$$em_{\text{queue n}}[Ethernetframes] = weight_{\text{queue n}}[Ethernetframes] \tag{7.1}$$

$em_{queue\ n}$: emission of egress port queue n in unit of Ethernet frames
$weight_{queue\ n}$: `EthSwtETSConfigAvailableBandwidthInWeightValue` configured for queue n in unit of Ethernet frames

One emission portion is equal to the configured emission of all neighboring egress port queues in Ethernet frames:

$$em_{\text{neighboring queues}}[Ethernetframes] = \sum_{n=1}^{N} em_{\text{queue n}}(n)[Ethernetframes] \tag{7.2}$$

N: count of neighboring egress port queues
$em_{neighboring\ queues}$: emission in unit of Ethernet frames of all neighboring egress port queues (one emission portion)
$em_{queue\ n}$: emission of egress port queue n in unit of Ethernet frames

⌋

[SWS_EthSwt_00548] Amount of Ethernet frames within one emission portion if available bandwidth is configured in weights as amount of Ethernet frames

Status: DRAFT
Upstream requirements: SRS_Eth_00121, SRS_Eth_00179, SRS_Eth_00180

⌈If an `EthSwtPortSchedulerAlgorithm` is configured with `ETHSWT_SCHEDULER_ENHANCED_TRAFFIC_SHAPER` and the `EthSwtPortEgressQueueTransmissionSelectionETSConfig` is set to `EthSwtETSConfigAvailableBandwidthInWeightValue` and all neighboring egress port queues offers an emission opportunity during the procession of one emission portion, then the emission portion shall enclose the amount of Ethernet frames of each neighboring egress port queue configured via `EthSwtETSConfigAvailableBandwidthInWeightValue`.⌋

[SWS_EthSwt_00549] Tolerance of egress port queue emission within the defined measurement interval if available bandwidth is configured in percent is used

Status: DRAFT
Upstream requirements: SRS_Eth_00121, SRS_Eth_00179, SRS_Eth_00180

⌈If an `EthSwtPortSchedulerAlgorithm` is configured with `ETHSWT_SCHEDULER_ENHANCED_TRAFFIC_SHAPER` and the `EthSwtPortEgressQueueTransmissionSelectionETSConfig` of all neighboring egress port queues is set to `EthSwtETSConfigAvailableBandwidthInPercent` and all neighboring egress port queues offers an emission opportunity during a defined measurement interval, then the emission of all egress port queues during this defined measurement interval shall reflect the configured bandwidth limitation of neighboring egress port queue in percent of the available bandwidth with a tolerance of 10 % (see [7, IEEE Std 802.1Q] chapter "ETS algorithm").⌋

The definition of a measurement interval need to consider the line rate (`EthSwtPortPhysicalLayerType`) of the according `EthSwtPort`. The `EthSwtPortPhysicalLayerType` defines the bit time. The measurement interval divided by bit time defines the amount of bits which is expected for the emission of all neighboring egress port queues.

[SWS_EthSwt_00550] Definition of bit time

Status: DRAFT
Upstream requirements: SRS_Eth_00121, SRS_Eth_00179, SRS_Eth_00180

⌈The time consumed to transmit a bit (bit time) shall be calculated according the following equation:

$$bit_{\text{time}}[\frac{s}{bit}] = \frac{1}{line_{\text{rate}}[\frac{Bit}{s}]} \tag{7.3}$$

$line_{rate}$[Bit/s]: `EthSwtPortPhysicalLayerType`⌋

[SWS_EthSwt_00551] Determination of egress port queue emission with available bandwidth configured in percent

Status: DRAFT
Upstream requirements: SRS_Eth_00121, SRS_Eth_00179, SRS_Eth_00180

⌈If the available bandwidth of neighboring egress port queues are configured with `EthSwtETSConfigAvailableBandwidthInPercent` and all egress port queues consume their available bandwidth, then the emission per egress port queue within the defined measurement interval of the according `EthSwtPortEgressScheduler` shall be determined by the following calculations:

The emission of all neighboring egress port queues in bits:

$$em_{\text{neighboring queues}}(T) = \frac{T[s]}{bit_{\text{time}}[\frac{s}{bit}]} \tag{7.4}$$

T: measurement interval in seconds
$em_{neighboring\ queues}$: emission in bits of all neighboring egress port queues per defined measurement interval

The emission of one egress queue in bits:

$$em_{\text{queue n}}(T)[bit] = em_{\text{neighboring queues}}(T)[bit] * \frac{bw_{\text{queue n}}}{100} \tag{7.5}$$

$em_{queue\ n}$: emission of egress port queue n in bits during the defined measurement interval
$bw_{queue\ n}$: bandwith of queue n in percent configured via `EthSwtETSConfigAvailableBandwidthInPercent`⌋

Note: If the available bandwidth of neighboring egress port queues is configured with `EthSwtETSConfigAvailableBandwidthInPercent`, then the total number of bits that are consumed on the medium by the transmission of the according Ethernet frames need to be considered for determining an emission, i.e. including all required framing bits like preamble, start frame delimiter (SFD), frame check sequence (FCS) and minimum inter-packet gap (IPG).

[SWS_EthSwt_CONSTR_00552] Constraint for configuration of available bandwidth in percent

Status: DRAFT

⌈If the available bandwidth of neighboring egress port queues configured with `EthSwtETSConfigAvailableBandwidthInPercent`, then the sum of the configured available bandwidth of all neigboring egress port queues shall result in 100 %.⌋

The available bandwidth of the neighboring egress port queues need to be shared on runtime, if egress port queues have bandwidth left over and their emission opertunity was resumed. A round-robin traffic shaping configured with `EthSwtPortEgressScheduler` set the `EthSwtPortSchedulerAlgorithm` to `ETHSWT_SCHEDULER_ENHANCED_TRAFFIC_SHAPER` should always try to utilize the complete available bandwidth.

[SWS_EthSwt_00553] Utilization of all neighboring egress port queues with a resumed emission opportunity

Status: DRAFT
Upstream requirements: SRS_Eth_00121, SRS_Eth_00179, SRS_Eth_00180

⌈Neighboring egress port queues with `EthSwtPortEgressQueueTransmissionSelectionAlgorithm` set to `ETHSWT_TRANSMISSION_SELECTION_ALGORITHM_ETS` shall equally share remaining available bandwidth on runtime. The sum of utilized bandwidth of all neighboring egress port queues with a resumed emission opportunity should result

- in the amount of Ethernet frames covered by one emission portion if available bandwidth is configured in weights as amount of Ethernet frames

- in approximately 100 % if available bandwidth is configured in percent

⌋

The available bandwidth per neighboring egress port queues could deviate on runtime from the configured available bandwidth (either in weights or in percent of the available bandwidth see (EthSwtPortEgressQueueTransmissionSelectionETSConfig)). The configured bandwidth represent the expected utilized bandwidth if all neighboring egress port queues consume their configured bandwidth within the defined measurement interval, i.e. all neighboring egress port queues resume the emission opportunity.

7.1.7.2.10 Transmission on the network

An Ethernet frame which is selected by the very last EthSwtPortEgressScheduler, will be transfered from the egress queue to the Ethernet network. As preparation for the transmission a so-called "Priority Code Point Encoding" and "VLAN forwarding" is performed. This is the last procession stage in the frame forwarding process. This processing stage ensure that all information of the Ethernet frame meta information are written in the Ethernet frame before the frame is forwared to an PHY. The Ethernet frame meta information contain the latest state of the Ethernet frame from the path through the Ethernet switch. An Example for the content of an Ethernet frame meta information:

- EthSwtPortPriorityRegenerationRegeneratedPriority: the Ethernet frame will be transmitted with this priority as VLAN-prioity in the VLAN-tag
- EthSwtVlanForwardingType set to ETHSWT_SENT_TAGGED: the Etherent frame will be transmitted with a VLAN-tag

[SWS_EthSwt_00554] Use regenerated priority value as VLAN-priority

Status: DRAFT
Upstream requirements: SRS_Eth_00121

⌈If an Ethernet frame is selected to be transmitted on the Ethernet network, then the Ethernet switch shall use the regenerated priority value of its meta information as VLAN-priority, if this Ethernet frame is transmitted with a VLAN-tag.⌋

7.1.7.3 Switch Management support

Switch Management enables the possibility to control an Ethernet frame regarding a Switch-Port specific ingress and egress handling as well as providing a 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 or path-delay measurement frame.

For an introduction of the basic HW architecture and interaction, please refer to [4, SWS_EthernetDriver].

[SWS_EthSwt_00240]

Upstream requirements: SRS_BSW_00171, SRS_Eth_00125

⌈The Switch driver shall offer Switch management APIs

- `EthSwt_EthRxProcessFrame`
- `EthSwt_EthRxFinishedIndication`
- `EthSwt_EthTxAdaptBufferLength`
- `EthSwt_EthTxPrepareFrame`
- `EthSwt_SetMgmtInfo`
- `EthSwt_EthTxProcessFrame` and
- `EthSwt_EthTxFinishedIndication`

if `EthSwtManagementSupportApi` is set to `TRUE`.⌋

Note: Switch management APIs support the EthIf to gather / modify Switch-Port specific communication attributes.

[SWS_EthSwt_00241]

Upstream requirements: SRS_Eth_00125

⌈The Switch Driver management APIs

- `EthSwt_EthRxProcessFrame`
- `EthSwt_EthRxFinishedIndication`
- `EthSwt_EthTxAdaptBufferLength`
- `EthSwt_EthTxPrepareFrame`
- `EthSwt_SetMgmtInfo`
- `EthSwt_EthTxProcessFrame` and
- `EthSwt_EthTxFinishedIndication`

shall support the Ethernet Driver to gather the Switch specific management information out of an Ethernet frame for reception or to prepare an Ethernet frame for management mode conformant frame transmission, e.g. the egress route of a frame.⌋

[SWS_EthSwt_00242]

Upstream requirements: SRS_Eth_00125

⌈The Switch Driver management APIs `EthSwt_EthTxProcessFrame` and `EthSwt_EthTxFinishedIndication` shall return immediately, if `EthSwt_SetMgmtInfo` has not been called before a call of `EthSwt_EthTxProcessFrame`.⌋

7.1.7.4 Global Time support

For more details regarding time measurement with Switches, please refer to [16, SWS_TimeSyncOverEthernet].

[SWS_EthSwt_00243]

Upstream requirements: SRS_BSW_00171, SRS_Eth_00125

⌈The Switch driver shall access the port specific hardware time stamps if EthSwtPortTimeStampSupport of the port is set to `TRUE`.⌋

[SWS_EthSwt_00378]

Upstream requirements: SRS_Eth_00125

⌈If `EthSwt_PortEnableTimeStamp` is called for a `PortIdx`, the switch driver shall enable the time-stamping for this port if `EthSwtPortTimeStampSupport` is set to `TRUE` for this port.⌋

7.1.7.5 Counter synchronization of Ethernet switches which are connected via uplink ports

Some Ethernet Switches provide the possibility to synchronize their internal clock. For Ethernet switches which are connected via uplink ports it is not necessary to measure the delay between the connected uplink ports, if the clock synchronization clock is activated (`EthSwtClockSynchronizationSupport` set to `TRUE`).

[SWS_EthSwt_00408] ⌈The Switch driver shall enable clock synchronization with another Ethernet switch to which it is connected via uplink port, if `EthSwtClockSynchronizationSupport` is set to `TRUE`.⌋

[SWS_EthSwt_CONSTR_00409] ⌈The port specific timestamping (EthSwtPortTimeStampSupport) can be set to `TRUE`, if clock synchronization for connected Ethernet switches is deactivated (`EthSwtClockSynchronizationSupport` set to `FALSE`).⌋

[SWS_EthSwt_CONSTR_00410] ⌈The port specific timestamping (`EthSwtPortTimeStampSupport`) can be set to `TRUE`, if `EthSwtClockSynchronizationSupport` is activated and `EthSwtPortRole` is not `ETHSWT_UP_LINK_PORT`. EthSwtPorts with `EthSwtPortRole` `ETHSWT_UP_LINK_PORT` are connected to another

Ethernet switch and not considered for the time delay compensation, if EthSwtClock-SynchronizationSupport is activated.⌋

7.1.7.6 Verification of Configuration

There are some situations where the Host controller needs to verify the Switch configuration.

[SWS_EthSwt_00292]

Upstream requirements: SRS_Eth_00126

⌈If the parameter EthSwtVerifyConfigApi is set to TRUE the function EthSwt_VerifyConfig shall be used to verify switch configuration.⌋

Implementation hint: As Switch configuration is highly HW-Architecture dependent the steps inside the function are implementation specific.

In some use cases, it is necessary to stop frame forwarding during the verification using the optional function EthSwt_SetForwardingMode

The function EthSwt_VerifyConfig could for example do the following steps:

- Stop frame forwarding by calling EthSwt_SetForwardingMode(FALSE).
- Verify the switch configuration
- In case the switch configuration is valid then frame forwarding shall be enabled by calling EthSwt_SetForwardingMode(TRUE) (if disabled in step 1).
- In case the switch configuration is not valid then the switch shall be reset and reconfigured.

Note: Please note that a reset of the Host Controller does not necessarily need a reset of the connected Switch HW. This needs to be evaluated individually very carefully as a reset raises the risk of uncontrolled communication during reset phase of the host controller.

Note: The Verification of the Switch Configuration as described above is just an example how and when this Verification may be done. It is very dependent on the used switch HW as well as the individual HW-Architecture and even Power supply and Reset strategy of the Switch of the ECU how the Configuration is verified or even how it can be verified. The only thing what this Module specifies is the interface to the upper layer to apply some verification on the switch configuration.

7.1.7.7 Testing and Diagnostic of Switch Ports

If configured, the Ethernet Switch Driver provides following interfaces to apply Testing and diagnostic functionalities

- EthSwt_GetPortSignalQuality
- EthSwt_GetPortIdentifier
- EthSwt_GetSwitchIdentifier
- EthSwt_WritePortMirrorConfiguration
- EthSwt_ReadPortMirrorConfiguration
- EthSwt_GetPortMirrorState
- EthSwt_SetPortMirrorState
- EthSwt_SetPortTestMode
- EthSwt_SetPortLoopbackMode
- EthSwt_SetPortTxMode
- EthSwt_GetPortCableDiagnosticsResult
- EthSwt_GetCfgDataRaw
- EthSwt_GetCfgDataInfo

The Availabiliy of these functions is strongly depending on the possibilities of the used Transceiver-(Phy)-HW.

7.1.7.8 Low Power Mode Support

[SWS_EthSwt_00376]

Upstream requirements: SRS_Eth_00183

⌈If EthSwtLowPowerModeSupport is set to TRUE and at least one EthSwtPort of a Ethernet switch is enabled and the corresponding Ethernet switch HW is in an inactive or low power mode the Ethernet switch HW shall be set to an active mode in which forwarding of Ethernet frames is possible.⌋

[SWS_EthSwt_00377]

Upstream requirements: SRS_Eth_00183

⌈If EthSwtLowPowerModeSupport is set to TRUE and no EthSwtPort for a certain Ethernet switch is enabled, the corresponding Ethernet switch HW shall be set to an inactive or low power mode.⌋

7.1.7.9 MACsec support

A valid MACsec configuration need to consider to exclude host and up link ports of an Ethernet switch from MACsec protection. Such ports have no direct link partner, which

runs an MKA instance as counter part to handle the key agreement protocol on a point to point connection.

[SWS_EthSwt_CONSTR_00616] Exclude host port and uplink ports from MACsec protection on a point to point connection ⌈A configuration of a EthSwtPort where the EthSwtPortMacSecEnabled is set to TRUE and EthSwtPortRole is configured (either as ETHSWT_HOST_PORT or ETHSWT_UP_LINK_PORT) shall be rejected as invalid.⌋

Some Ethernet traffic needs to be handled as unprotected traffic (e.g. MKPDUs for key agreement). Therefore the Ethernet switch driver provide the possibility at its EthSwtPorts to configure bypass rules to bypass Ethernet frames that match the configured rules (e.g. EthSwtPortMacSecBypassEtherType). Multiple rules can be configured at the same EthSwtPort. Those rules are considered as OR-linked list. For example, if EthSwtPortMacSecBypassEtherType is set to 0x888E (EAPOL Ethernet type) and EthSwtPortMacSecBypassVlan set to 233, then all Ethernet frames with Ethernet type 0x888E are bypassed (independent of the VLAN-ID) and all Ethernet frames with VLAN-ID set to 233 are bypassed (independent of the Ethernet type).

The configured bypass rules need to be downloaded to the corresponding Ethernet switch port as soon as the Ethernet switch driver gets initialized.

[SWS_EthSwt_00564] Initialize Ethernet switch ports with MACsec support, where MACsec functionality is enabled

Upstream requirements: FO_RS_MACsec_00004

⌈At initialization of an Ethernet switch, the EthSwt driver shall initialize each Ethernet switch port with the corresponding EthSwtPort configuration where EthSwtPortMacSecEnabled is available and set to TRUE, and consider the following configuration parameter if available:

- initialize the Ethernet switch port with all configured EthSwtPortMacSecBypassEtherTypes to bypass MACsec handling for all Ethernet frames, that carrying one of the configured EtherTypes
- initialize the Ethernet switch port with all configured EthSwtPortMacSecBypassVlans to bypass MACsec handling for all Ethernet frames, that carrying one of the configured VLAN-IDs
- initialize the Ethernet switch port with all configured EthSwtPortMacSecBypassDestinationMacAddresses to bypass MACsec handling for all Ethernet frames, that carrying one of the configured destination MAC addresses

⌋

[SWS_EthSwt_00565] Initialize Ethernet switch ports with MACsec support, where MACsec functionality is disabled

Upstream requirements: FO_RS_MACsec_00004

⌈At initialization of an Ethernet switch, the EthSwt driver shall consider the following points for each Ethernet switch port where the corresponding EthSwtPort configuration has EthSwtPortMacSecEnabled set to FALSE:

- ignore all MACsec related configurations (e.g. EthSwtPortMacSecBypassEtherTypes)
- ensure to put the Ethernet port to a state, where all Ethernet frames are handled without processing of MACsec functionality

⌋

A secured association for reception and transmission is created via call of EthSwt_MacSecAddRxSa and EthSwt_MacSecAddTxSa. The API calls control via the input parameter Active if a secured association is active after creation (Active = TRUE) or inactive after creation (Active = FALSE). A call with EthSwt_MacSecUpdateRxSa and EthSwt_MacSecUpdateTxSa would activate (Active = TRUE) or deactivate (Active = FALSE) an existing secured association given with An. At the same time only one secured association is allowed to exist per association number given with An. Before a new secured association is added to a specific An, the previous secured association identified with this An need to be deleted for reception via a call of EthSwt_MacSecDeleteRxSa and for transmission via a call of EthSwt_MacSecDeleteTxSa.

[SWS_EthSwt_00566] Handling if EthSwt_MacSecInitRxSc is called and EthSwtPortMacSecEnabled is available and set to TRUE

Status: DRAFT
Upstream requirements: FO_RS_MACsec_00004

⌈If EthSwt_MacSecInitRxSc is called and EthSwtPortMacSecEnabled is available and set to TRUE of the EthSwtPort identified with the given SwitchPortIdx at the EthSwtConfig identified with the given SwitchIdx, then the EthSwt driver shall configure a receive secure channel with the secure channel id given with Sci at the Ethernet switch port given with SwitchPortIdx at the Ethernet switch that corresponds to the given SwitchIdx and return with E_OK.⌋

[SWS_EthSwt_00567] Handling if EthSwt_MacSecInitRxSc is called and EthSwtPortMacSecEnabled is set to FALSE or the configuration parameter is not available

Status: DRAFT
Upstream requirements: FO_RS_MACsec_00004

⌈If EthSwt_MacSecInitRxSc is called and EthSwtPortMacSecEnabled is set to FALSE or the configuration parameter is not available at the EthSwtPort identified with the given SwitchPortIdx at the EthSwtConfig identified with the given SwitchIdx, then the EthSwt driver shall return with E_NOT_OK.⌋

[SWS_EthSwt_00568] Handling if `EthSwt_MacSecUpdateSecY` is called and `EthSwtPortMacSecEnabled` is available and set to `TRUE`

Status: DRAFT
Upstream requirements: FO_RS_MACsec_00004

⌈If `EthSwt_MacSecUpdateSecY` is called and `EthSwtPortMacSecEnabled` is available and set to `TRUE` of the `EthSwtPort` identified with the given `SwitchPortIdx` at the `EthSwtConfig` identified with the given `SwitchIdx`, then the EthSwt driver shall consider to update the Ethernet switch port that corresponds to the given `SwitchPortIdx` at the Ethernet switch that corresponds to the given `SwitchIdx` with the MAC security entity configuration provided via `MACsecCfgPtr` and consider to establish a transmit secure channel with the transmission secure channel id given with `TxSci` and return with `E_OK`.⌋

[SWS_EthSwt_00569] Handling if attempts to update an Ethernet switch port with a MAC security entity configuration and the establishment of a transmit secure channel has been finalized

Status: DRAFT
Upstream requirements: FO_RS_MACsec_00004

⌈If `EthSwt_MacSecUpdateSecY` was called for an `EthSwtPort` with `EthSwtPortMacSecEnabled` set to `TRUE` and the update process for the corresponding Ethernet switch port with the MAC security entity configuration and the establishment of a transmit secure channel has been successfully finalized, then the EthSwt driver shall notify the EthIf by calling `EthIf_SwitchMacSecUpdateSecYNotification` with `Result` set to `E_OK`. Otherwise the EthSwt driver shall call `EthIf_SwitchMacSecUpdateSecYNotification` with `Result` set to `E_NOT_OK`.⌋

Note: A retry to update an Ethernet switch port with an MAC security entity configuration and to establish a transmit secure channel is in responsibility of the north bound layer.

[SWS_EthSwt_00570] Handling if `EthSwt_MacSecUpdateSecY` is called and update of a MAC security entity configuration and the establishment of a transmit secure channel was already successfully finalized

Status: DRAFT
Upstream requirements: FO_RS_MACsec_00004

⌈If `EthSwt_MacSecUpdateSecY` is called and the following conditions are valid:

- `EthSwtPortMacSecEnabled` is available and set to `TRUE` of the `EthSwtPort` identified with the given `SwitchPortIdx` at the `EthSwtConfig` identified with the given `SwitchIdx`
- the Ethernet switch port was already successfully updated with of a MAC security entity configuration given with `MACsecCfgPtr`
- the establishment of a transmit secure channel with the given `TxSci` was already successfully finalized

then the EthSwt driver shall consider to update the Ethernet switch port that corresponds to the given SwitchPortIdx at the Ethernet switch that corresponds to the given SwitchIdx with the MAC security entity configuration provided via MACsecCfgPtr and return with E_OK.⌋

Note: According to the given MACsec configuration to update an Ethernet switch port, it could be possible to disable MACsec handling completely at the given Ethernet switch port and to release the established transmit secure channel.

[SWS_EthSwt_00571] Handling if EthSwt_MacSecUpdateSecY is called and EthSwtPortMacSecEnabled is set to FALSE or the configuration parameter is not available

Status: DRAFT
Upstream requirements: FO_RS_MACsec_00004

⌈If EthSwt_MacSecUpdateSecY is called and EthSwtPortMacSecEnabled is set to FALSE or the configuration parameter is not available at the EthSwtPort identified with the given SwitchPortIdx at the EthSwtConfig identified with the given SwitchIdx, then the EthSwt driver shall return with E_NOT_OK.⌋

[SWS_EthSwt_00572] Handling if EthSwt_MacSecResetRxSc is called and EthSwtPortMacSecEnabled is available and set to TRUE

Status: DRAFT
Upstream requirements: FO_RS_MACsec_00004

⌈If EthSwt_MacSecResetRxSc is called and the following conditions are valid:

- EthSwtPortMacSecEnabled is available and set to TRUE of the EthSwtPort identified with the given SwitchPortIdx at the EthSwtConfig identified with the given SwitchIdx
- a receive secure channel with the given Sci exists at this Ethernet switch port

then the EthSwt driver shall delete the receive secure channel given with Sci at the Ethernet switch port given with SwitchPortIdx at the Ethernet switch that corresponds to the given SwitchIdx and return with E_OK. Otherwise the EthSwt driver shall return with E_NOT_OK.⌋

[SWS_EthSwt_00573] Handling if EthSwt_MacSecResetRxSc is called and EthSwtPortMacSecEnabled is set to FALSE or the configuration parameter is not available

Status: DRAFT
Upstream requirements: FO_RS_MACsec_00004

⌈If EthSwt_MacSecResetRxSc is called and EthSwtPortMacSecEnabled is set to FALSE or the configuration parameter is not available at the EthSwtPort identified with the given SwitchPortIdx at the EthSwtConfig identified with the given SwitchIdx, then the EthSwt driver shall return with E_NOT_OK.⌋

[SWS_EthSwt_00574] Handling if EthSwt_MacSecAddRxSa is called and EthSwtPortMacSecEnabled is available and set to TRUE

Status: DRAFT
Upstream requirements: FO_RS_MACsec_00004

⌈If EthSwt_MacSecAddRxSa is called and the following conditions are valid:

- EthSwtPortMacSecEnabled is available and set to TRUE of the EthSwtPort identified with the given SwitchPortIdx at the EthSwtConfig identified with the given SwitchIdx
- a reception secure association can be added to the given association number (see An), i.e. the given association number is not occupied

then the EthSwt driver shall perform the following actions. Otherwise return with E_NOT_OK:

- create a reception secure association at the Ethernet switch port that corresponds to the given SwitchPortIdx at the Ethernet switch that corresponds to the given SwitchIdx by considering the given association number (see An), the lowest acceptable packet number (see LowestPn), the short secure channel identifier (see Ssci) and the secured association key (available via KeysPtr and addressed with SakKeyPtr)
- activate the created reception secure association, if given API parameter Active is set to TRUE, otherwise deactivate reception secure association
- return with E_OK.

⌋

[SWS_EthSwt_00575] Handling if EthSwt_MacSecAddRxSa is called and EthSwtPortMacSecEnabled is set to FALSE or the configuration parameter is not available

Status: DRAFT
Upstream requirements: FO_RS_MACsec_00004

⌈If EthSwt_MacSecAddRxSa is called and EthSwtPortMacSecEnabled is set to FALSE or the configuration parameter is not available at the EthSwtPort identified with the given SwitchPortIdx at the EthSwtConfig identified with the given SwitchIdx, then the EthSwt driver shall return with E_NOT_OK.⌋

[SWS_EthSwt_00576] Handling if attempts to add a reception secure association at an Ethernet switch port and to enable or disable this secure association has been finalized

Status: DRAFT
Upstream requirements: FO_RS_MACsec_00004

⌈If EthSwt_MacSecAddRxSa was called for an EthSwtPort with EthSwtPortMacSecEnabled set to TRUE and the attempts to add a reception secure association

and to enable or disable this reception secure association for the corresponding Ethernet switch port has been successfully finalized, then the EthSwt driver shall notify the EthIf by calling EthIf_SwitchMacSecAddRxSaNotification with Result set to E_OK. Otherwise the EthSwt driver shall call EthIf_SwitchMacSecAddRxSaNotification with Result set to E_NOT_OK.⌋

[SWS_EthSwt_00577] Handling if EthSwt_MacSecUpdateRxSa is called with Active is set to TRUE, the secured association is deactivated and EthSwtPortMacSecEnabled is available and set to TRUE

Status: DRAFT
Upstream requirements: FO_RS_MACsec_00004

⌈If EthSwt_MacSecUpdateRxSa is called and the following conditions are valid:

- EthSwtPortMacSecEnabled is available and set to TRUE of the EthSwtPort identified with the given SwitchPortIdx at the EthSwtConfig identified with the given SwitchIdx
- the given API parameter Active is set to TRUE
- the reception secured association given with An exists and is deactivated

then the EthSwt driver shall update the reception secure association at the Ethernet switch port that corresponds to the given SwitchPortIdx at the Ethernet switch that corresponds to the given SwitchIdx by considering the lowest acceptable packet number (see LowestPn) and activate the reception secure association at this Ethernet switch port and return with E_OK. Otherwise, if the addressed reception secured association does not exist, return with E_NOT_OK.⌋

[SWS_EthSwt_00578] Handling if EthSwt_MacSecUpdateRxSa is called with Active is set to TRUE, the secured association is activated and EthSwtPortMacSecEnabled is available and set to TRUE

Status: DRAFT
Upstream requirements: FO_RS_MACsec_00004

⌈If EthSwt_MacSecUpdateRxSa is called and the following conditions are valid:

- EthSwtPortMacSecEnabled is available and set to TRUE of the EthSwtPort identified with the given SwitchPortIdx at the EthSwtConfig identified with the given SwitchIdx
- the given API parameter Active is set to TRUE
- the reception secured association given with An exists and has already been activated

then the EthSwt driver shall update the reception secure association at the Ethernet switch port that corresponds to the given SwitchPortIdx at the Ethernet switch that corresponds to the given SwitchIdx by considering the lowest acceptable packet number (see LowestPn) and return with E_OK. Otherwise, if the addressed reception secured association does not exist, return with E_NOT_OK⌋

[SWS_EthSwt_00579] Handling if `EthSwt_MacSecUpdateRxSa` is called with `Active` is set to `FALSE`, the secured association is activated and `EthSwtPortMacSecEnabled` is available and set to `TRUE`

Status: DRAFT
Upstream requirements: FO_RS_MACsec_00004

⌈If `EthSwt_MacSecUpdateRxSa` is called and the following conditions are valid:

- `EthSwtPortMacSecEnabled` is available and set to `TRUE` of the `EthSwtPort` identified with the given `SwitchPortIdx` at the `EthSwtConfig` identified with the given `SwitchIdx`
- the given API parameter `Active` is set to `FALSE`
- the reception secured association given with `An` exists and is activated

then the EthSwt driver shall deactivate the reception secure association with the given with `An` at the Ethernet switch port that corresponds to the given `SwitchPortIdx` at the Ethernet switch that corresponds to the given `SwitchIdx` by considering the lowest acceptable packet number (see `LowestPn`) and return with `E_OK`. Otherwise, if the addressed reception secured association given with `An` does not exist, return with `E_NOT_OK`⌋

[SWS_EthSwt_00580] Handling if `EthSwt_MacSecUpdateRxSa` is called with `Active` is set to `FALSE`, the secured association is deactivated and `EthSwtPortMacSecEnabled` is available and set to `TRUE`

Status: DRAFT
Upstream requirements: FO_RS_MACsec_00004

⌈If `EthSwt_MacSecUpdateRxSa` is called and the following conditions are valid:

- `EthSwtPortMacSecEnabled` is available and set to `TRUE` of the `EthSwtPort` identified with the given `SwitchPortIdx` at the `EthSwtConfig` identified with the given `SwitchIdx`
- the given API parameter `Active` is set to `FALSE`
- the reception secured association given with `An` exists and has already been deactivated

then the EthSwt driver shall ignore the API call and return with `E_OK`. Otherwise, if the addressed reception secured association given with `An` does not exist, return with `E_NOT_OK`⌋

[SWS_EthSwt_00581] Handling if EthSwt_MacSecUpdateRxSa is called and EthSwtPortMacSecEnabled is set to FALSE or the configuration parameter is not available

Status: DRAFT
Upstream requirements: FO_RS_MACsec_00004

⌈If EthSwt_MacSecUpdateRxSa is called and EthSwtPortMacSecEnabled is set to FALSE or the configuration parameter is not available at the EthSwtPort identified with the given SwitchPortIdx at the EthSwtConfig identified with the given SwitchIdx, then the EthSwt driver shall return with E_NOT_OK.⌋

[SWS_EthSwt_00582] Handling if EthSwt_MacSecDeleteRxSa is called and EthSwtPortMacSecEnabled is available and set to TRUE

Status: DRAFT
Upstream requirements: FO_RS_MACsec_00004

⌈If EthSwt_MacSecDeleteRxSa is called and the following conditions are valid:

- EthSwtPortMacSecEnabled is available and set to TRUE of the EthSwtPort identified with the given SwitchPortIdx at the EthSwtConfig identified with the given SwitchIdx
- a reception secure association with the given An exists at this Ethernet switch port

then the EthSwt driver shall remove the reception secure association given with An at the Ethernet switch port given with SwitchPortIdx at the Ethernet switch that corresponds to the given SwitchIdx and return with E_OK.Otherwise, if addressed secured association given with An is not available, the EthSwt driver shall return with E_NOT_OK.⌋

[SWS_EthSwt_00583] Handling if EthSwt_MacSecDeleteRxSa is called and EthSwtPortMacSecEnabled is set to FALSE or the configuration parameter is not available

Status: DRAFT
Upstream requirements: FO_RS_MACsec_00004

⌈If EthSwt_MacSecDeleteRxSa is called and EthSwtPortMacSecEnabled is set to FALSE or the configuration parameter is not available at the EthSwtPort identified with the given SwitchPortIdx at the EthSwtConfig identified with the given SwitchIdx, then the EthSwt driver shall return with E_NOT_OK.⌋

[SWS_EthSwt_00584] Handling if EthSwt_MacSecAddTxSa is called and EthSwtPortMacSecEnabled is available and set to TRUE

Status: DRAFT
Upstream requirements: FO_RS_MACsec_00004

⌈If EthSwt_MacSecAddTxSa is called and the following conditions are valid:

- EthSwtPortMacSecEnabled is available and set to TRUE of the EthSwtPort identified with the given SwitchPortIdx at the EthSwtConfig identified with the given SwitchIdx

- a transmission secure association can be added to the given association number (see An), i.e. the given association number is not occupied

then the EthSwt driver shall perform the following actions. Otherwise return with E_NOT_OK:

- create a transmission secure association at the Ethernet switch port that corresponds to the given SwitchPortIdx at the Ethernet switch that corresponds to the given SwitchIdx by considering the given association number (see An), the next accepted packet number (see NextPn), the short secure channel identifier (see Ssci) and the secured association key (available via KeysPtr and addressed with SakKeyPtr)
- activate the created transmission secure association, if given API parameter Active is set to TRUE, otherwise deactivate the created transmission secure association
- return with E_OK

⌋

[SWS_EthSwt_00585] Handling if EthSwt_MacSecAddTxSa is called and EthSwtPortMacSecEnabled is set to FALSE or the configuration parameter is not available

Status: DRAFT
Upstream requirements: FO_RS_MACsec_00004

⌈If EthSwt_MacSecAddTxSa is called and EthSwtPortMacSecEnabled is set to FALSE or the configuration parameter is not available at the EthSwtPort identified with the given SwitchPortIdx at the EthSwtConfig identified with the given SwitchIdx, then the EthSwt driver shall return with E_NOT_OK.⌋

[SWS_EthSwt_00586] Handling if attempts to enable / disable the transmission secure association at an Ethernet switch port has been finalized

Status: DRAFT
Upstream requirements: FO_RS_MACsec_00004

⌈If EthSwt_MacSecAddTxSa was called for an EthSwtPort with EthSwtPortMacSecEnabled set to TRUE and attempts to add a transmission secure association and to enable or disable of this transmission secure association for the corresponding Ethernet switch port has been successfully finalized, then the EthSwt driver shall notify the EthIf by calling EthIf_SwitchMacSecAddTxSaNotification with SwitchIdx of MgmtInfoPtr set to to the corresponding EthIfSwitchIdx and SwitchIdx of MgmtInfoPtr set to the EthSwtPortIdx and Result set to E_OK. Otherwise the EthSwt driver shall call EthIf_SwitchMacSecAddTxSaNotification with Result set to E_NOT_OK.⌋

[SWS_EthSwt_00587] Handling if `EthSwt_MacSecUpdateTxSa` is called with `Active` is set to `TRUE`, the secured association is deactivated and `EthSwtPortMacSecEnabled` is available and set to `TRUE`

Status: DRAFT
Upstream requirements: FO_RS_MACsec_00004

⌈If `EthSwt_MacSecUpdateTxSa` is called and the following conditions are valid:

- `EthSwtPortMacSecEnabled` is available and set to `TRUE` of the `EthSwtPort` identified with the given `SwitchPortIdx` at the `EthSwtConfig` identified with the given `SwitchIdx`
- the given API parameter `Active` is set to `TRUE`
- the transmission secured association given with `An` exists and is deactivated

then the EthSwt driver shall perform the following actions

- update the transmission secure association at the Ethernet switch port that corresponds to the given `SwitchPortIdx` at the Ethernet switch that corresponds to the given `SwitchIdx` by considering the next packet number (see `NextPn`)
- activate the transmission secure association given with `An` at this Ethernet switch port
- if another transmission secured association at this Ethernet switch port is running, then disable this secured association
- return with `E_OK`

Otherwise, if the addressed transmission secured association given with `An` does not exist, return with `E_NOT_OK`⌋

[SWS_EthSwt_00588] Handling if `EthSwt_MacSecUpdateTxSa` is called with `Active` is set to `TRUE`, the secured association is activated and `EthSwtPortMacSecEnabled` is available and set to `TRUE`

Status: DRAFT
Upstream requirements: FO_RS_MACsec_00004

⌈If `EthSwt_MacSecUpdateTxSa` is called and the following conditions are valid:

- `EthSwtPortMacSecEnabled` is available and set to `TRUE` of the `EthSwtPort` identified with the given `SwitchPortIdx` at the `EthSwtConfig` identified with the given `SwitchIdx`
- the given API parameter `Active` is set to `TRUE`
- the transmission secured association given with `An` exists and has already been activated

then the EthSwt driver shall update the transmission secure association given with `An` at the Ethernet switch port that corresponds to the given `SwitchPortIdx` at the Ethernet switch that corresponds to the given `SwitchIdx` by considering the next packet

number (see NextPn) and return with E_OK. Otherwise, if the addressed transmission secured association given with An does not exist, return with E_NOT_OK⌋

[SWS_EthSwt_00589] Handling if EthSwt_MacSecUpdateTxSa is called with Active is set to FALSE, the secured association is activated and EthSwtPortMacSecEnabled is available and set to TRUE

Status: DRAFT
Upstream requirements: FO_RS_MACsec_00004

⌈If EthSwt_MacSecUpdateTxSa is called and the following conditions are valid:

- EthSwtPortMacSecEnabled is available and set to TRUE of the EthSwtPort identified with the given SwitchPortIdx at the EthSwtConfig identified with the given SwitchIdx
- the given API parameter Active is set to FALSE
- the transmission secured association given with An exists and is activated

then the EthSwt driver shall deactivate the transmission secure association with the given with An at the Ethernet switch port that corresponds to the given SwitchPortIdx at the Ethernet switch that corresponds to the given SwitchIdx by considering the next accepted packet number (see NextPn) and return with E_OK. Otherwise, if the addressed transmission secured association given with An does not exist, return with E_NOT_OK⌋

[SWS_EthSwt_00590] Handling if EthSwt_MacSecUpdateTxSa is called with Active is set to FALSE, the secured association is deactivated and EthSwtPortMacSecEnabled is available and set to TRUE

Status: DRAFT
Upstream requirements: FO_RS_MACsec_00004

⌈If EthSwt_MacSecUpdateTxSa is called and the following conditions are valid:

- EthSwtPortMacSecEnabled is available and set to TRUE of the EthSwtPort identified with the given SwitchPortIdx at the EthSwtConfig identified with the given SwitchIdx
- the given API parameter Active is set to FALSE
- the transmission secured association given with An exists and has already been deactivated

then the EthSwt driver shall ignore the API call and return with E_OK. Otherwise, if the addressed transmission secured association does not exist, return with E_NOT_OK⌋

[SWS_EthSwt_00591] Handling if EthSwt_MacSecUpdateTxSa is called and EthSwtPortMacSecEnabled is set to FALSE or the configuration parameter is not available

Status: DRAFT
Upstream requirements: FO_RS_MACsec_00004

⌈If EthSwt_MacSecUpdateTxSa is called and EthSwtPortMacSecEnabled is set to FALSE or the configuration parameter is not available at the EthSwtPort identified with the given SwitchPortIdx at the EthSwtConfig identified with the given SwitchIdx, then the EthSwt driver shall return with E_NOT_OK.⌋

[SWS_EthSwt_00592] Handling if EthSwt_MacSecDeleteTxSa is called and EthSwtPortMacSecEnabled is available and set to TRUE

Status: DRAFT
Upstream requirements: FO_RS_MACsec_00004

⌈If EthSwt_MacSecDeleteTxSa is called and the following conditions are valid:

- EthSwtPortMacSecEnabled is available and set to TRUE of the EthSwtPort identified with the given SwitchPortIdx at the EthSwtConfig identified with the given SwitchIdx
- a transmission secure association with the given An exists at this Ethernet switch port

then the EthSwt driver shall remove the transmission secure association given with An at the Ethernet switch port given with SwitchPortIdx at the Ethernet switch that corresponds to the given SwitchIdx and return with E_OK.Otherwise, if addressed secured association is not available, the EthSwt driver shall return with E_NOT_OK.⌋

[SWS_EthSwt_00593] Handling if EthSwt_MacSecDeleteTxSa is called and EthSwtPortMacSecEnabled is set to FALSE or the configuration parameter is not available

Status: DRAFT
Upstream requirements: FO_RS_MACsec_00004

⌈If EthSwt_MacSecDeleteTxSa is called and EthSwtPortMacSecEnabled is set to FALSE or the configuration parameter is not available at the EthSwtPort identified with the given SwitchPortIdx at the EthSwtConfig identified with the given SwitchIdx, then the EthSwt driver shall return with E_NOT_OK.⌋

[SWS_EthSwt_00594] Handling if EthSwt_MacSecGetTxSaNextPn is called and EthSwtPortMacSecEnabled is available and set to TRUE

Status: DRAFT
Upstream requirements: FO_RS_MACsec_00004

⌈If EthSwt_MacSecGetTxSaNextPn is called and the following conditions are valid:

- EthSwtPortMacSecEnabled is available and set to TRUE of the EthSwtPort identified with the given SwitchPortIdx at the EthSwtConfig identified with the given SwitchIdx

- the transmission secured association given with An exists

then the EthSwt driver shall return the next package number from the MACsec entity (SecY) at the Ethernet switch port that corresponds to the given SwitchPortIdx at the Ethernet switch that corresponds to the given SwitchIdx via the out parameter NextPnPtr and return with E_OK. Otherwise, if the addressed transmission secured association does not exist, return with E_NOT_OK⌋

[SWS_EthSwt_00595] Handling if EthSwt_MacSecGetTxSaNextPn is called and EthSwtPortMacSecEnabled is set to FALSE or the configuration parameter is not available

Status: DRAFT
Upstream requirements: FO_RS_MACsec_00004

⌈If EthSwt_MacSecGetTxSaNextPn is called and EthSwtPortMacSecEnabled is set to FALSE or the configuration parameter is not available at the EthSwtPort identified with the given SwitchPortIdx at the EthSwtConfig identified with the given SwitchIdx, then the EthSwt driver shall return with E_NOT_OK.⌋

[SWS_EthSwt_00596] Handling if EthSwt_MacSecGetMacSecStatistics is called and EthSwtPortMacSecEnabled is available and set to TRUE

Status: DRAFT
Upstream requirements: FO_RS_MACsec_00004

⌈If EthSwt_MacSecGetMacSecStatistics is called and the EthSwtPortMacSecEnabled is available and set to TRUE of the EthSwtPort identified with the given SwitchPortIdx at the EthSwtConfig identified with the given SwitchIdx, then the EthSwt driver shall consider to collect the MACsec statistics of this EthSwtPort and return with E_OK.⌋

[SWS_EthSwt_00597] Handling if EthSwt_MacSecGetMacSecStatistics is called and EthSwtPortMacSecEnabled is set to FALSE or the configuration parameter is not available

Status: DRAFT
Upstream requirements: FO_RS_MACsec_00004

⌈If EthSwt_MacSecGetMacSecStatistics is called and EthSwtPortMacSecEnabled is set to FALSE or the configuration parameter is not available at the EthSwtPort identified with the given SwitchPortIdx at the EthSwtConfig identified with the given SwitchIdx, then the EthSwt driver shall return with E_NOT_OK.⌋

[SWS_EthSwt_00598] Handling if attempts to collect MACsec statistics at an Ethernet switch port has been finalized

Status: DRAFT
Upstream requirements: FO_RS_MACsec_00004

⌈If EthSwt_MacSecGetMacSecStatistics was called for an EthSwtPort with EthSwtPortMacSecEnabled set to TRUE and the collection of the MACsec statistics for the corresponding Ethernet switch port has been successfully finalized, then the EthSwt driver shall perform the following actions:

- set all statistics value which are not available (e.g. not provided by the hardware) to the maximum value of the used data type (e.g. for uint64 set value to 0xFFFFFFFFFFFFFFFF)
- notify the EthIf by calling `EthIf_SwitchMacSecGetMacSecStatistics-Notification` with `SwitchIdx` of `MgmtInfoPtr` set to to the corresponding `EthIfSwitchIdx` and `SwitchPortIdx` of `MgmtInfoPtr` set to the `EthSwt-PortIdx`

⌋

[SWS_EthSwt_00599] Handling if `EthSwt_MacSecSetControlledPortEnabled` is called and `EthSwtPortMacSecEnabled` is available and set to `TRUE`

Status: DRAFT
Upstream requirements: FO_RS_MACsec_00004

⌈If `EthSwt_MacSecSetControlledPortEnabled` is called and the `EthSwtPort-MacSecEnabled` is available and set to `TRUE` of the `EthSwtPort` identified with the given `SwitchPortIdx` at the `EthSwtConfig` identified with the given `SwitchIdx`, then the EthSwt driver shall consider to set the PAE of this `EthSwtPort` to the activation state (either enabled (`TRUE`) or disabled (`FALSE`)) given with `ControlledPort-Enabled` and return with `E_OK`.⌋

[SWS_EthSwt_00600] Handling if `EthSwt_MacSecSetControlledPortEnabled` is called and `EthSwtPortMacSecEnabled` is set to `FALSE` or the configuration parameter is not available

Status: DRAFT
Upstream requirements: FO_RS_MACsec_00004

⌈If `EthSwt_MacSecSetControlledPortEnabled` is called and `EthSwtPortMac-SecEnabled` is set to `FALSE` or the configuration parameter is not available at the `EthSwtPort` identified with the given `SwitchPortIdx` at the `EthSwtConfig` identified with the given `SwitchIdx`, then the EthSwt driver shall return with `E_NOT_OK`.⌋

7.2 Error Classifications

Chapter [10, General Specification of Basic Software Modules] 7.2 *"Error Handling"* describes the error handling of the Basic Software in detail. Above all, it constitutes a classification scheme consisting of following error types which may occur in BSW modules.

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

7.2.1 Development Errors

[SWS_EthSwt_00001] Definition of development errors in module EthSwt

Upstream requirements: SRS_BSW_00385

⌈

Type of error	*Related error code*	*Error value*
Invalid switch index	ETHSWT_E_INV_SWITCH_IDX	0x01
EthSwt module was not initialized	ETHSWT_E_UNINIT	0x02
Invalid pointer in parameter list	ETHSWT_E_PARAM_POINTER	0x03
Invalid API which is not available by another module	ETHSWT_E_INV_API	0x05
Invalid switch port index	ETHSWT_E_INV_SWITCHPORT_IDX	0x06
Invalid Controller Index	ETHSWT_E_INV_CTRL_IDX	0x07
Invalid input parameter	ETHSWT_E_INV_PARAM	0x08
Invalid configuration	ETHSWT_E_INIT_FAILED	0x09

⌋

[SWS_EthSwt_00009]

Upstream requirements: SRS_BSW_00323, SRS_BSW_00369

⌈If development error detection is enabled, the function `EthSwt_Init` shall check the parameter `CfgPtr` for being valid. If the check fails, `EthSwt_Init` shall raise the development error `ETHSWT_E_INIT_FAILED`.⌋

Note: Please note that in case of variant pre-compile `NULL_PTR` is allowed.

[SWS_EthSwt_00164]

Upstream requirements: SRS_BSW_00369, SRS_BSW_00386, SRS_Eth_00118

⌈The switch driver shall check whether the lower layer driver, i.e. the EthTrcv provides the APIs which can be called by an upper layer module (EthIf) of the switch driver and will be forwarded to the lower layer. In case of missing APIs, the switch driver shall raise the development error `ETHSWT_E_INV_API` if APIs are missing in the lower layer module.⌋

Note: This check will be performed upon calling a certain API. For this check the input parameter `SwitchPortIdx` and a configuration table which needs to be derived from the configuration of the Ethernet transceiver drivers which are attached to the Ethernet switch driver are necessary. This functionality is necessary if development error tracing is activated. This check is necessary because an Ethernet switch driver API can be called by an upper layer module with the argument `SwitchPortIdx`. This value of this `SwitchPortIdx` can be in a valid range, but some Ethernet transceiver driver which are used by the switch driver support the API and some do not support this API. In order to resolve this conflict, this check has been implemented.

[SWS_EthSwt_00156]

Upstream requirements: SRS_BSW_00413, SRS_BSW_00323, SRS_BSW_00369, SRS_Eth_00118

⌈The function EthSwt_SetSwitchPortMode shall check whether the EthTrcv_SetTransceiverMode API of the indexed transceiver driver is available by checking whether for this SwitchPortIdx the corresponding EthTrcv API is available. If this is not the case, the function shall raise the development error ETHSWT_E_INV_API.⌋

[SWS_EthSwt_00157]

Upstream requirements: SRS_BSW_00413, SRS_BSW_00323, SRS_BSW_00369, SRS_Eth_00118

⌈The function EthSwt_GetSwitchPortMode shall check whether the EthTrcv_GetTransceiverMode API of the indexed transceiver driver is available by checking whether for this SwitchPortIdx the corresponding EthTrcv API is available. If this is not the case, the function shall raise the development error ETHSWT_E_INV_API.⌋

[SWS_EthSwt_00386]

Upstream requirements: SRS_BSW_00350

⌈If development error detection is activated by EthSwtDevErrorDetect, all functions except EthSwt_Init shall check that the service EthSwt_Init was previously called. If the check fails, the function shall raise the development error ETHSWT_E_UNINIT.⌋

[SWS_EthSwt_00387]

Upstream requirements: SRS_BSW_00350

⌈If development error detection is activated by EthSwtDevErrorDetect, all functions with input parameter SwitchIdx shall check the parameter for being valid. If the check fails, the functions shall raise the development error ETHSWT_E_INV_SWITCH_IDX.⌋

[SWS_EthSwt_00389]

Upstream requirements: SRS_BSW_00350

⌈If development error detection is enabled, all functions with input parameter SwitchPortIdx or PortIdx shall check the parameter for being valid. If the check fails, the functions shall raise the development error ETHSWT_E_INV_SWITCH_IDX.⌋

[SWS_EthSwt_00390]

Upstream requirements: SRS_BSW_00350

⌈If development error detection is enabled, all functions with input parameter CtrlIdx shall check the parameter for being valid. If the check fails, the functions shall raise the development error ETHSWT_E_INV_CTRL_IDX.⌋

[SWS_EthSwt_00391]

Upstream requirements: SRS_BSW_00350

⌈If development error detection is enabled, all functions with input parameter `BufIdx` shall check the parameter for being valid. If the check fails, the functions shall raise the development error `ETHSWT_E_INV_PARAM`.⌋

[SWS_EthSwt_00392]

Upstream requirements: SRS_BSW_00350

⌈If development error detection is enabled, all functions with inout or output pointer parameter shall check the parameter for being valid. If the check fails, the functions shall raise the development error `ETHSWT_E_PARAM_POINTER`.⌋

[SWS_EthSwt_00393]

Upstream requirements: SRS_BSW_00350

⌈If development error tracing is activated by `EthSwtDevErrorDetect`, the functions which call an Ethernet Transceiver API and do not obtain the functionality directly from the switch port interface shall check whether the API of the indexed transceiver driver is available. If this is not the case, the functions shall raise the development error `ETHSWT_E_INV_API`.⌋

[SWS_EthSwt_00154]

Upstream requirements: SRS_Eth_00118, SRS_Eth_00119, SRS_BSW_00413, SRS_BSW_00323, SRS_BSW_00369

⌈If development error detection is activated by `EthSwtDevErrorDetect`, the function `EthSwt_GetLinkState` shall check whether the `EthTrcv_GetLinkState` API of the indexed transceiver driver is available by checking whether for this `SwitchPortIdx` the corresponding `EthTrcv` API is available. If this is not the case, the function shall raise the development error `ETHSWT_E_INV_API`.⌋

7.2.2 Runtime Errors

[SWS_EthSwt_00434] Definition of runtime errors in module EthSwt ⌈

Type of error	*Related error code*	*Error value*
Initialization of ports is not finished	ETHSWT_INIT_NOT_COMPLETED	0x01

⌋

7.2.3 Transient Faults

There are no transient faults.

7.2.4 Production Errors

[SWS_EthSwt_00113] Ethernet Switch Access Failure

Upstream requirements: SRS_BSW_00385

⌈

Diagnostic Event (Error Name)	ETHSWT_E_ACCESS
Description	This production error shall be issued when the switch is not accessible.
Failed condition	When access to the Ethernet Switch fails the module shall report the production error with event status DEM_EVENT_STATUS_PREFAILED to DEM.
Passed condition	When access to the Ethernet Switch succeeds the module shall report the production error with event status DEM_EVENT_STATUS_PREPASSED to DEM.

⌋

[SWS_EthSwt_00395] Ethernet switch port and the referenced Ethernet transceiver are in contradicting modes.

Upstream requirements: SRS_BSW_00385

⌈

Diagnostic Event (Error Name)	ETHSWT_E_SYNCPORT2PHY
Description	While getting the Ethernet switch port mode, the Ethernet switch driver detected an inconsistent state between Ethernet switch port and the referenced Ethernet transceiver Mode.
Failed condition	When getting the Ethernet switch port mode together with the Ethernet transceiver mode and the mode of the two referenced modules was found inconsistent the module shall report the production error with event status DEM_EVENT_STATUS_ PREFAILED to DEM.
Passed condition	When getting the Ethernet switch port mode together with the Ethernet transceiver mode and the mode of the two referenced modules was found consistent the module shall report the production error with event status DEM_EVENT_STATUS_ PREPASSED to DEM.

⌋

8 API specification

8.1 Imported types

This chapter lists all types included from the following files:

[SWS_EthSwt_00002] Definition of imported datatypes of module EthSwt ⌈

Module	*Header File*	*Imported Type*
Comtype	ComStack_Types.h	TimeStampType (draft)
Dem	Rte_Dem_Type.h	Dem_EventIdType
	Rte_Dem_Type.h	Dem_EventStatusType
Eth	Eth_GeneralTypes.h	Eth_BufIdxType
	Eth_GeneralTypes.h	Eth_CounterType
	Eth_GeneralTypes.h	Eth_MacVlanType
	Eth_GeneralTypes.h	Eth_ModeType
	Eth_GeneralTypes.h	Eth_RxStatsType
	Eth_GeneralTypes.h	Eth_StreamStatisticCounterType
	Eth_GeneralTypes.h	Eth_TxErrorCounterValuesType
	Eth_GeneralTypes.h	Eth_TxStatsType
EthTrcv	Eth_GeneralTypes.h	EthTrcv_BaudRateType
	Eth_GeneralTypes.h	EthTrcv_CableDiagResultType
	Eth_GeneralTypes.h	EthTrcv_DuplexModeType
	Eth_GeneralTypes.h	EthTrcv_LinkStateType
	Eth_GeneralTypes.h	EthTrcv_PhyLoopbackModeType
	Eth_GeneralTypes.h	EthTrcv_PhyTestModeType
	Eth_GeneralTypes.h	EthTrcv_PhyTxModeType
	Eth_GeneralTypes.h	EthTrcv_WakeupReasonType
Mka	Mka.h	Mka_ConfidentialityOffsetType (draft)
	Mka.h	Mka_MacSecConfigType (draft)
	Mka.h	Mka_SakKeyPtrType (draft)
	Mka.h	Mka_Stats_Rx_ScType (draft)
	Mka.h	Mka_Stats_Rx_SecYType (draft)
	Mka.h	Mka_Stats_SecYType (draft)
	Mka.h	Mka_Stats_Tx_ScType (draft)
	Mka.h	Mka_Stats_Tx_SecYType (draft)
	Mka.h	Mka_ValidateFramesType (draft)
NvM	Rte_NvM_Type.h	NvM_BlockIdType
	Rte_NvM_Type.h	NvM_BlockRequestType
	Rte_NvM_Type.h	NvM_RequestResultType
Spi	Spi.h	Spi_AsyncModeType
	Spi.h	Spi_ChannelType
	Spi.h	Spi_DataBufferType

▽

△

Module	*Header File*	*Imported Type*
	Spi.h	Spi_NumberOfDataType
	Spi.h	Spi_SequenceType
Std	Std_Types.h	Std_ReturnType

⌋

8.2 Type definitions

8.2.1 EthSwt_StateType

[SWS_EthSwt_00123] Definition of datatype EthSwt_StateType

Upstream requirements: SRS_BSW_00406

⌈

Name	EthSwt_StateType		
Kind	Enumeration		
Range	ETHSWT_STATE_UNINIT	0x00	Switch is not yet configured
	ETHSWT_STATE_INIT	0x01	Switch driver is initialized
	ETHSWT_STATE_ PORTINIT_COMPLETED	0x02	Port initialization is completed
	ETHSWT_STATE_ACTIVE	0x03	Switch is active
Description	Status supervision used for Development Error Detection. The state shall be available for debugging.		
Available via	Eth_GeneralTypes.h		

⌋

8.2.2 EthSwt_ConfigType

[SWS_EthSwt_00165] Definition of datatype EthSwt_ConfigType

Upstream requirements: SRS_BSW_00395

⌈

Name	EthSwt_ConfigType	
Kind	Structure	
Elements	implementation specific	
	Type	–
	Comment	–
Description	Implementation specific structure of the post build configuration.	
Available via	EthSwt.h	

⌋

8.2.3 EthSwt_MacLearningType

[SWS_EthSwt_00227] Definition of datatype EthSwt_MacLearningType

Upstream requirements: SRS_Eth_00087

⌈

Name	EthSwt_MacLearningType		
Kind	Enumeration		
Range	ETHSWT_ MACLEARNING_ HWDISABLED	–	If hardware learning disabled, the switch must not learn new MAC addresses
	ETHSWT_ MACLEARNING_ HWENABLED	–	If hardware learning enabled, the switch learns new MAC addresses
	ETHSWT_ MACLEARNING_ SWENABLED	–	If software learning enabled, the hardware learning is disabled and the switch forwards packets with an unknown source address to a host CPU
Description	The interpretation of this value		
Available via	Eth_GeneralTypes.h		

⌋

8.2.4 EthSwt_MgmtInfoType

[SWS_EthSwt_91002] Definition of datatype EthSwt_MgmtInfoType

Upstream requirements: SRS_Eth_00125

⌈

Name	EthSwt_MgmtInfoType	
Kind	Structure	
Elements	SwitchIdx	
	Type	uint8
	Comment	Switch index
	SwitchPortIdx	
	Type	uint8
	Comment	Port index of the switch
Description	Type for holding the management information received/transmitted on Switches (ports).	
Available via	Eth_GeneralTypes.h	

⌋

8.2.5 EthSwt_PortMirrorCfgType

[SWS_EthSwt_91017] Definition of datatype EthSwt_PortMirrorCfgType

Upstream requirements: SRS_Eth_00123

⌈

Name	EthSwt_PortMirrorCfgType	
Kind	Structure	
Elements	srcMacAddrFilter	
	Type	Array of uint8
	Size	6
	Comment	Specifies the source MAC address [0..255,0..255,0..255,0..255,0..255,0..255] that should be mirrored. If set to 0,0,0,0,0,0, no source MAC address filtering shall take place.
	dstMacAddrFilter	
	Type	Array of uint8
	Size	6
	Comment	Specifies the destination MAC address [0..255,0..255,0..255,0..255,0..255,0..255] that should be mirrored. If set to 0,0,0,0,0,0, no destination MAC address filtering shall take place.
	VlanIdFilter	
	Type	uint16
	Comment	Specifies the VLAN address 0..4094 that should be mirrored. If set to 65535, no VLAN filtering shall take place.
	MirroringPacketDivider	
	Type	uint8
	Comment	Divider if only a subset of received frames should be mirrored. E.g. MirroringPacketDivider = 2 means every second frames is mirrored
	MirroringMode	
	Type	uint8
	Comment	specifies the mode how the mirrored traffic should be tagged : 0x00 == No VLAN retagging; 0x01 == VLAN retagging; 0x02 == VLAN Double tagging
	TrafficDirectionIngressBitMask	
	Type	uint32
	Comment	Specifies the bit mask of Ethernet switch ingress port traffic direction to be mirrored. The bit mask is calculated depending of the values of EthSwtPortIdx. (e.g. set EthSwtPortIdx == 2 => TrafficDirectionIngressBitMask = 0b0000 0000 0000 0000 0000 0000 0000 0100). 0b0 == enable ingress port mirroring 0b1 == disable ingress port mirroring Example: TrafficDirectionIngressBitMask = 0b0000 0000 0000 0000 0000 0000 0000 0100 => Ingress traffic mirroring is enabled of Ethernet switch port with EthSwtPortIdx=2
	TrafficDirectionEgressBitMask	
	Type	uint32

▽

△

	Comment	Specifies the bit mask of Ethernet switch egress port traffic direction to be mirrored. The bit mask is calculated depending of the values of Eth SwtPortIdx. (e.g. set EthSwtPortIdx == 2 => TrafficDirectionEgressBit Mask = 0b0000 0000 0000 0000 0000 0000 0000 0100). 0b0 == enable egress port mirroring 0b1 == disable egress port mirroring Example: TrafficDirectionEgressBitMask = 0b0000 0000 0000 0000 0000 0000 0000 0001 => Egress traffic mirroring is enabled of Ethernet switch port with EthSwtPortIdx=0
	CapturePortIdx	
	Type	uint8
	Comment	Specifies the Ethernet switch port which capture the mirrored traffic
	ReTaggingVlanId	
	Type	uint16
	Comment	Specifies the VLAN address 0..4094 which shall be used for re-tagging if MirroringMode is set to 0x01 (VLAN re-tagging). If the value is set to 65535, the value shall be ignored, because the VLAN address for re-tagging is provided by the Ethernet switch configuration
	DoubleTaggingVlanId	
	Type	uint16
	Comment	Specifies the VLAN address 0..4094 which shall be used for double-tagging if MirroringMode is set to 0x02 (VLAN double tagging). If the value is set to 65535, the value shall be ignored, because the VLAN address for double tagging is provided by the Ethernet switch configuration
Description	The EthSwt_PortMirrorCfgType specify the port mirror configuration which is set up per Ethernet switch. The configuration is written to the Ethernet switch driver by calling EthSwt_WritePortMirror Configuration. One port mirror configuration is maintained per Ethernet Switch.	
Available via	Eth_GeneralTypes.h	

⌋

8.2.6 EthSwt_PortMirrorStateType

[SWS_EthSwt_91020] Definition of datatype EthSwt_PortMirrorStateType

Upstream requirements: SRS_Eth_00123

⌈

Name	EthSwt_PortMirrorStateType		
Kind	Enumeration		
Range	PORT_MIRRORING_ DISABLED	0x00	port mirroring disabled
	PORT_MIRRORING_ ENABLED	0x01	port mirroring enabled
Description	Type to request or obtain the port mirroring state (enable/disable) for a particular port mirror configuration per Ethernet switch.		
Available via	Eth_GeneralTypes.h		

⌋

8.2.7 EthSwt_ReturnType

[SWS_EthSwt_91033] Definition of Std_ReturnType-extension for module EthSwt ⌈

Range	ETHSWT_PORT_ MIRRORING_ CONFIGURATION_NOT_ SUPPORTED	0x02	port mirroring configuration is not supported by Ethernet switch driver or by the Ethernet switch hardware
Description	Overlayed return value of Std_ReturnType for Ethernet switch driver API EthSwt_WritePortMirror Configuration, if the port mirroring configuration is not supported by Ethernet switch driver or by the Ethernet switch hardware (e.g. the configured mirrored traffic direction (see [SWS_EthSwt_91017] "TrafficDirectionIngressBitMask" and "TrafficDirectionEgressBitMask") for ingress and egress traffic of the same port is not supported, or the addressed Ethernet switch ports within the port mirror configuration are not accessible by the Ethernet switch driver)		
Available via	Eth_GeneralTypes.h		

⌋

8.2.8 EthSwt_MgmtOwner

[SWS_EthSwt_91035] Definition of datatype EthSwt_MgmtOwner ⌈

Name	EthSwt_MgmtOwner		
Kind	Enumeration		
Range	ETHSWT_MGMT_OBJ_ UNUSED	0x00	Object unused
	ETHSWT_MGMT_OBJ_ OWNED_BY_ETHSWT	0x01	Object used and EthSwt collects needed data
	ETHSWT_MGMT_OBJ_ OWNED_BY_UPPER_ LAYER	0x02	Object used and the upper layer does calculations
Description	Holds information if upper layer or EthSwt is owner of mgmt_obj.		
Available via	Eth_GeneralTypes.h		

⌋

8.2.9 EthSwt_Mgmt_ObjectType

[SWS_EthSwt_91037] Definition of datatype EthSwt_MgmtObjectType ⌈

Name	EthSwt_MgmtObjectType	
Kind	Structure	
Elements	Validation	
	Type	EthSwt_MgmtObjectValidType
	Comment	The validation information for the mgmt_obj.
	IngressTimestamp	
	Type	TimeStampType
	Comment	The ingress timestamp value out of the switch.

▽

△

	EgressTimestamp	
	Type	TimeStampType
	Comment	The egress timestamp value out of the switch.
	MgmtInfo	
	Type	EthSwt_MgmtInfoType
	Comment	Received/Transmitted Management information of the switches.
	Ownership	
	Type	EthSwt_MgmtOwner
	Comment	The ownership of MgmtObj.
Description	Provides information about all struct member elements. The ownership gives information whether EthSwt has finished its activities in providing all struct member elements.	
Available via	Eth_GeneralTypes.h	

⌋

[SWS_EthSwt_00433] ⌈A MgmtObject is just allowed to be owned between EthSwt and only one <UPPER_LAYER>. The structure element can be identified unambiguously using the DataPtr in Rx- and BufIdx in Tx-context, because both elements are definitively unique within the RxIndication() / TxConfirmation() context.⌋

8.2.10 EthSwt_MgmtObjectValidType

[SWS_EthSwt_91036] Definition of datatype EthSwt_MgmtObjectValidType ⌈

Name	EthSwt_MgmtObjectValidType	
Kind	Structure	
Elements	IngressTimestampValid	
	Type	Std_ReturnType
	Comment	IngressTimestampValid shall be set to E_NOT_OK if ingress timestamp is not available
	EgressTimestampValid	
	Type	Std_ReturnType
	Comment	EgressTimestampValid shall be set to E_NOT_OK if ingress timestamp is not available.
	MgmtInfoValid	
	Type	Std_ReturnType
	Comment	MgmtInfoValid shall be set to E_NOT_OK if ingress timestamp is not available(e.g. timeout).
Description	Will be set from EthSwt and marks EthSwt_MgmtObject as valid or not. So the upper layer will be able to detect inconsistencies.	
Available via	Eth_GeneralTypes.h	

⌋

8.3 Function definitions

This chapter describes the API functions provided by the module

8.3.1 Generic functions

8.3.1.1 EthSwt_Init

[SWS_EthSwt_00006] Definition of API function EthSwt_Init

Upstream requirements: SRS_BSW_00101

⌈

Service Name	EthSwt_Init	
Syntax	`void EthSwt_Init ( const EthSwt_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 Switch Driver	
Available via	EthSwt.h	

⌋

[SWS_EthSwt_00007]

Upstream requirements: SRS_BSW_00101

⌈The function `EthSwt_Init` shall store the access to the configuration structure for subsequent API calls.⌋

[SWS_EthSwt_00008]

Upstream requirements: SRS_BSW_00101

⌈The function `EthSwt_Init` shall change the state of all switches controlled by this Switch Driver from `ETHSWT_STATE_UNINIT` to `ETHSWT_STATE_INIT`.⌋

[SWS_EthSwt_00421]

Upstream requirements: SRS_Eth_00123

⌈The EthSwt shall check for enabled port mirror configuration. The enabled port mirror configuration shall be activated by reconfiguring the Ethernet switch hardware according to the port mirror configuration, before frame forwarding is being enabled.⌋

[SWS_EthSwt_00422]

Upstream requirements: SRS_Eth_00123

⌈If the PortMirrorState is set to 0x01 (port mirroring enabled), then the stored port mirror configuration for the given Ethernet switch shall be written to hardware registers of the given Ethernet switch and enable port mirroring.⌋

[SWS_EthSwt_00423] ⌈If the PortMirrorState is set to 0x00 (port mirroring disabled) the corresponding hardware registers of the given Ethernet switch shall be reset (to the HW's default values) and the port mirroring shall be disabled.⌋

[SWS_EthSwt_00011]

Upstream requirements: SRS_BSW_00101

⌈After initialization of the Ethernet switch within the `EthSwt_BackgroundTask`, the Ethernet switch shall enter an inactive or low power mode if `EthSwtLowPowerModeSupport` is set to TRUE. If `EthSwtLowPowerModeSupport` is not defined or set to FALSE the Ethernet switch shall enter an active state.⌋

Note: The execution of this function may take a long time (e.g. port structure, VLAN configuration, internal Ethernet switch engine ... a.s.o.) and therefore cannot be called by EcuM or BswM. Instead it should be called e.g. by a background task (see `EthSwt_BackgroundTask`).

[SWS_EthSwt_00374] ⌈All Ethernet switch HW ports which are not configured as a EthSwtPort shall be switched off during initialization. This Ethernet switch HW ports shall never be switched on during runtime⌋

[SWS_EthSwt_00375] ⌈All EthSwtPorts shall be set to `ETH_MODE_DOWN` during initialization.⌋

[SWS_EthSwt_00016]

Upstream requirements: SRS_BSW_00386

⌈The function `EthSwt_Init` shall check the access to the Ethernet Switch hardware, i.e. by trying to read or write registers during the configuration of the switch. If the access to the registers fails, the function shall raise the extended production error `ETHSWT_E_ACCESS` and return `E_NOT_OK`.⌋

Note: Access to the Ethernet Switch hardware is device dependent, e.g. access through the Ethernet Controller Mii, access through SPI, ... etc.

8.3.1.2 EthSwt_GetSwitchIdentifier

[SWS_EthSwt_91016] Definition of API function EthSwt_GetSwitchIdentifier

Upstream requirements: SRS_Eth_00123

⌈

Service Name	EthSwt_GetSwitchIdentifier	
Syntax	`Std_ReturnType EthSwt_GetSwitchIdentifier ( uint8 SwitchIdx, uint32* OrgUniqueIdPtr )`	
Service ID [hex]	0x35	
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	EthSwt.h	

⌋

[SWS_EthSwt_00305]

Upstream requirements: SRS_Eth_00123

⌈The function `EthSwt_GetSwitchIdentifier` shall return the value of the organizationally unique identifier of the indexed Ethernet switch.⌋

[SWS_EthSwt_00308]

Upstream requirements: SRS_BSW_00171

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

8.3.1.3 EthSwt_SetForwardingMode

[SWS_EthSwt_91013] Definition of API function EthSwt_SetForwardingMode

Upstream requirements: SRS_Eth_00126

⌈

Service Name	EthSwt_SetForwardingMode	
Syntax	`Std_ReturnType EthSwt_SetForwardingMode ( uint8 SwitchIdx, boolean mode )`	
Service ID [hex]	0x32	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	SwitchIdx	Index of the switch within the context of the Ethernet Switch Driver
	mode	True Forewarding 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	Configures switch to start or stop forwarding for all ports. This API call may be used during switch configuration verification.	
Available via	EthSwt.h	

⌋

[SWS_EthSwt_00291]

Upstream requirements: SRS_BSW_00171

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

8.3.1.4 EthSwt_CheckWakeup

[SWS_EthSwt_91003] Definition of API function EthSwt_CheckWakeup

Upstream requirements: SRS_Eth_00118

⌈

Service Name	EthSwt_CheckWakeup	
Syntax	`Std_ReturnType EthSwt_CheckWakeup ( uint8 SwitchIdx )`	
Service ID [hex]	0x4c	
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	Std_ReturnType	`E_OK`: request to check for a wake-up is accepted `E_NOT_OK`: request to check for a wake-up is not accepted
Description	API is called by EthIf. The Ethernet switch driver request to check for a wake-up at all Ethernet switch ports which reference an EthTrcv. For those Ethernet switch ports the call is forwarded to the referenced EthTrcv. The function could be called in context of an interrupt service routine or on task level Note: Interrupt service routine consuming time has to be considered, since all EthSwtPorts of the maintained Ethernet switches has to be checked. Therefore the call is forwarded to the referred EthTrcv where the request to check for wake-up is stored. The check of the Ethernet hardware is done asynchronously in the context of the EthTrcv_MainFunction.	
Available via	EthSwt.h	

⌋

[SWS_EthSwt_00440]

Upstream requirements: SRS_Eth_00118

⌈The function `EthSwt_CheckWakeup` shall iterate over the Ethernet switch ports of the indexed Ethernet switch and forward the call to `EthTrcv_CheckWakeup` for those Ethernet switch ports, which reference an EthTrcv using the driver API abstraction of the EthIf.⌋

[SWS_EthSwt_00441]

Upstream requirements: SRS_BSW_00171

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

8.3.2 Switch Configuration

8.3.2.1 EthSwt_VerifyConfig

[SWS_EthSwt_91012] Definition of API function EthSwt_VerifyConfig

Upstream requirements: SRS_Eth_00126

⌈

Service Name	EthSwt_VerifyConfig	

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

Service ID [hex]	0x31	
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)	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	Verifies the Switch Configuration depending on the HW-Architecture, HW-capability and the intended accuracy of this verification.	
Available via	EthSwt.h	

⌋

[SWS_EthSwt_00287]

Upstream requirements: SRS_BSW_00171

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

8.3.2.2 EthSwt_StoreConfiguration

[SWS_EthSwt_00086] Definition of API function EthSwt_StoreConfiguration

Upstream requirements: SRS_Eth_00087, SRS_Eth_00122

⌈

Service Name	EthSwt_StoreConfiguration	
Syntax	`Std_ReturnType EthSwt_StoreConfiguration ( uint8 SwitchIdx )`	
Service ID [hex]	0x13	
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 store the MAC/Port table was accepted `E_NOT_OK`: Request to persistently store the MAC/Port table was 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	EthSwt.h	

⌋

[SWS_EthSwt_00090]

Upstream requirements: SRS_BSW_00171

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

8.3.2.3 EthSwt_ResetConfiguration

[SWS_EthSwt_00091] Definition of API function EthSwt_ResetConfiguration

Upstream requirements: SRS_Eth_00087, SRS_Eth_00122

⌈

Service Name	EthSwt_ResetConfiguration	
Syntax	`Std_ReturnType EthSwt_ResetConfiguration ( uint8 SwitchIdx )`	
Service ID [hex]	0x14	
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 and store 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.	
Available via	EthSwt.h	

⌋

[SWS_EthSwt_00095]

Upstream requirements: SRS_BSW_00171

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

8.3.2.4 EthSwt_NvmSingleBlockCallback

[SWS_EthSwt_00125] Definition of callback function EthSwt_NvmSingleBlock Callback

Upstream requirements: SRS_Eth_00087, SRS_Eth_00122

⌈

Service Name	EthSwt_NvmSingleBlockCallback
Syntax	`Std_ReturnType EthSwt_NvmSingleBlockCallback ( NvM_BlockRequestType BlockRequest, NvM_RequestResultType JobResult )`
Service ID [hex]	0x17
Sync/Async	Synchronous
Reentrancy	Non Reentrant

▽

△

Parameters (in)	BlockRequest	The request type (read, write, ... etc.) of the previous processed block job
	JobResult	Covers the job result of the previous processed single block job.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: success `E_NOT_OK`: Callback function has not been processed successfully
Description	Function will be called by the NVRAMManager after the switch configuration has been stored or resetted.	
Available via	EthSwt_NvM.h	

⌋

[SWS_EthSwt_00126]

Upstream requirements: SRS_Eth_00122, SRS_Eth_00087

⌈The function `EthSwt_NvmSingleBlockCallback` shall be called by the NVRAM-Manager [17] after the switch configuration has been stored or reset in the the NV RAM.⌋

[SWS_EthSwt_00196]

Upstream requirements: SRS_Eth_00122, SRS_Eth_00087

⌈The function `EthSwt_NvmSingleBlockCallback` shall call the function <user>_PersistentConfigurationResult to provide the `JobResult` to the caller of `EthSwt_StoreConfiguration` or `EthSwt_ResetConfiguration`.⌋

[SWS_EthSwt_00127]

Upstream requirements: SRS_Eth_00122, SRS_Eth_00087

⌈The function `EthSwt_NvmSingleBlockCallback` shall always return `E_OK` according to `SWS_NvM_00368`.⌋

[SWS_EthSwt_00128]

Upstream requirements: SRS_BSW_00369, SRS_BSW_00472

⌈The function `EthSwt_NvmSingleBlockCallback` shall raise a development error if the `JobResult` equals `NVM_REQ_NOT_OK`, i.e. the write request has been finished unsuccessfully.⌋

Note: Please note that a production error at this point is not necessary because the NvM will raise also a production error if the write to NV RAM was not successful.

[SWS_EthSwt_00129]

Upstream requirements: SRS_BSW_00171

⌈The function `EthSwt_NvmSingleBlockCallback` shall be pre compile time configurable On/Off by the existence of the container `EthSwtNvm`.⌋

8.3.2.5 EthSwtPersistentConfigurationResultCallback

[SWS_EthSwt_00193] Definition of callback function <EthSwtPersistentConfigurationResultCallback>

Upstream requirements: SRS_Eth_00122, SRS_Eth_00087

⌈

Service Name	<EthSwtPersistentConfigurationResultCallback>	
Syntax	`void <EthSwtPersistentConfigurationResultCallback> ( NvM_RequestResultType JobResult )`	
Service ID [hex]	0x1b	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	JobResult	Covers the job result of the previous processed single block job.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Job end notification of EthSwt_StoreConfiguration or EthSwt_ResetConfiguration	
Available via	EthSwtExternals.h	

⌋

[SWS_EthSwt_00194]

Upstream requirements: SRS_Eth_00122, SRS_Eth_00087

⌈The callback function <EthSwtPersistentConfigurationResultCallback> shall be called by the `EthSwt_NvmSingleBlockCallback` to inform the caller of `EthSwt_StoreConfiguration` or `EthSwt_ResetConfiguration` about the state of the past calls.⌋

8.3.2.6 EthSwt_GetCfgDataRaw

[SWS_EthSwt_91030] Definition of API function EthSwt_GetCfgDataRaw

Upstream requirements: SRS_Eth_00123

⌈

Service Name	EthSwt_GetCfgDataRaw
Syntax	`Std_ReturnType EthSwt_GetCfgDataRaw ( uint8 SwitchIdx, uint32 Offset, uint16 Length, uint8* BufferPtr )`
Service ID [hex]	0x41
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	EthSwt.h	

⌋

[SWS_EthSwt_00403]

Upstream requirements: SRS_BSW_00171

⌈The function `EthSwt_GetCfgDataRaw` shall only be available if parameter EthSwtGetCfgRaw is set to TRUE.⌋

[SWS_EthSwt_00404] ⌈When calling the function `EthSwt_GetCfgDataRaw`, the function shall check the access to the Ethernet switch driver. If the check fails, the function shall raise the extended production error `ETHSWT_E_ACCESS` and return `E_NOT_OK`, otherwise pass the extended production error `ETHSWT_E_ACCESS` and return `E_OK`.⌋

8.3.2.7 EthSwt_GetCfgDataInfo

[SWS_EthSwt_91031] Definition of API function EthSwt_GetCfgDataInfo

Upstream requirements: SRS_Eth_00123

⌈

Service Name	EthSwt_GetCfgDataInfo	
Syntax	`Std_ReturnType EthSwt_GetCfgDataInfo ( uint8 SwitchIdx, uint32* DataSizePtr, uint32* DataAdressPtr )`	
Service ID [hex]	0x42	
Sync/Async	Synchronous	
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	EthSwt.h	

⌋

[SWS_EthSwt_00405]

Upstream requirements: SRS_BSW_00171

⌈The function `EthSwt_GetCfgDataInfo` shall only be available if parameter EthSwtGetCfgRaw is set to TRUE.⌋

[SWS_EthSwt_00406] ⌈When calling the function `EthSwt_GetCfgDataInfo`, the function shall check the access to the Ethernet switch driver. If the check fails, the function shall raise the extended production error `ETHSWT_E_ACCESS` and return `E_NOT_OK`, otherwise pass the extended production error `ETHSWT_E_ACCESS` and return `E_OK`.⌋

8.3.3 Port Mode

8.3.3.1 EthSwt_SetSwitchPortMode

[SWS_EthSwt_00018] Definition of API function EthSwt_SetSwitchPortMode

Upstream requirements: SRS_Eth_00118

⌈

Service Name	EthSwt_SetSwitchPortMode	
Syntax	`Std_ReturnType EthSwt_SetSwitchPortMode ( uint8 SwitchIdx, uint8 SwitchPortIdx, Eth_ModeType PortMode )`	
Service ID [hex]	0x03	
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

▽

△

	PortMode	ETH_MODE_DOWN: Disable the addressed Ethernet switch port at the given Ethernet switch ETH_MODE_ACTIVE: Enable the addressed Ethernet switch port at the given Ethernet switch ETH_MODE_ACTIVE_WITH_WAKEUP_REQUEST: Enable the addressed Ethernet switch port at the given Ethernet switch and request to trigger a wake-up on the network. (This could be used e.g. for Ethernet hardware which is compatible with the OA TC10)
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: success E_NOT_OK: The indexed switch port could not be set to Port Mode, or the function is called in state ETHSWT_STATE_UNINIT or ETHSWT_STATE_INIT.
Description	Enables/disables the indexed switch port	
Available via	EthSwt.h	

⌋

[SWS_EthSwt_00019]

Upstream requirements: SRS_Eth_00118

⌈The function EthSwt_SetSwitchPortMode shall put the indexed port of the switch into the specified mode. If EthSwtPort references an EthTrcv then the function EthTrcv_SetTransceiverMode of the Ethernet Transceiver Driver shall additionally be called with the corresponding transceiver mode using the driver API abstraction of the EthIf.⌋

[SWS_EthSwt_00396] ⌈When calling the function EthSwt_SetSwitchPortMode with mode ETH_MODE_DOWN, the EthSwt shall disable the Ethernet switch port directly for reduction of power consumption, if it is possible.⌋

[SWS_EthSwt_00397] ⌈When calling the function EthSwt_SetSwitchPortMode, the function shall check the access to the Ethernet switch driver. If the check fails, the function shall raise the extended production error ETHSWT_E_ACCESS and return E_NOT_OK, otherwise pass the extended production error ETHSWT_E_ACCESS and return E_OK.⌋

[SWS_EthSwt_00398]

Upstream requirements: SRS_Eth_00118

⌈If EthSwtPort does not references an EthTrcv, EthSwt shall indicate a mode of the port by the API EthIf_SwitchPortModeIndication latest during the next EthSwt_MainFunction.⌋

[SWS_EthSwt_00022]

Upstream requirements: SRS_BSW_00171

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

[SWS_EthSwt_00023]

Upstream requirements: SRS_Eth_00118

⌈If the switch is already in the requested mode E_OK shall be returned and no development error shall be raised.⌋

8.3.3.2 EthSwt_GetSwitchPortMode

[SWS_EthSwt_00025] Definition of API function EthSwt_GetSwitchPortMode

Upstream requirements: SRS_Eth_00118

⌈

Service Name	EthSwt_GetSwitchPortMode	
Syntax	`Std_ReturnType EthSwt_GetSwitchPortMode ( uint8 SwitchIdx, uint8 SwitchPortIdx, Eth_ModeType* SwitchModePtr )`	
Service ID [hex]	0x04	
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)	SwitchModePtr	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	EthSwt.h	

⌋

[SWS_EthSwt_00026]

Upstream requirements: SRS_Eth_00118

⌈The function EthSwt_GetSwitchPortMode shall read the mode of the indexed port of the switch. If EthSwtPort references an EthTrcv then the function shall additionally call the corresponding function EthTrcv_GetTransceiverMode of the Ethernet Transceiver Driver using the driver API abstraction of the EthIf.⌋

[SWS_EthSwt_00439] ⌈The function shall report the active mode always as ETH_MODE_ACTIVE, even though the previous requested (via EthSwt_SetSwitchPortMode) mode was ETH_MODE_ACTIVE_WITH_WAKEUP_REQUEST.⌋

[SWS_EthSwt_00399] ⌈If the obtained modes of the EthSwtPort and the EthTrcv are not aligned, the function EthSwt_GetSwitchPortMode shall raise the extended production error ETHSWT_E_SYNCPORT2PHY and return E_NOT_OK.

If EthTrcv_GetTransceiverMode returns E_NOT_OK, the EthSwt_GetSwitchPortMode shall also return E_NOT_OK without raising an error.⌋

[SWS_EthSwt_00400] ⌈If the function EthSwt_GetSwitchPortMode is called, the function shall check the access to the Ethernet Switch Driver. If the check fails, the function shall raise the extended production error ETHSWT_E_ACCESS and return E_NOT_OK, otherwise pass the production error ETHSWT_E_ACCESS and return E_OK.⌋

[SWS_EthSwt_00029]

Upstream requirements: SRS_BSW_00171

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

8.3.3.3 EthSwt_StartSwitchPortAutoNegotiation

[SWS_EthSwt_00031] Definition of API function EthSwt_StartSwitchPortAutoNegotiation

Upstream requirements: SRS_Eth_00087

⌈

Service Name	EthSwt_StartSwitchPortAutoNegotiation	
Syntax	`Std_ReturnType EthSwt_StartSwitchPortAutoNegotiation ( uint8 SwitchIdx, uint8 SwitchPortIdx )`	
Service ID [hex]	0x05	
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)	None	
Return value	Std_ReturnType	E_OK: success E_NOT_OK: Automatic negotiation could not be started for the indexed switch port, or the function is called in state ETHSWT_STATE_UNINIT or ETHSWT_STATE_INIT.
Description	Starts the auto-negotiation of the indexed switch port	
Available via	EthSwt.h	

⌋

[SWS_EthSwt_00032]

Upstream requirements: SRS_Eth_00087

⌈The function EthSwt_StartSwitchPortAutoNegotiation shall restart the automatic negotiation of the used transmission parameters of the referenced Ethernet transceiver driver by calling the function EthTrcv_StartAutoNegotiation using the driver API abstraction of the EthIf.⌋

[SWS_EthSwt_00035]

Upstream requirements: SRS_BSW_00171

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

8.3.3.4 EthSwt_GetSwitchPortWakeupReason

[SWS_EthSwt_91040] Definition of API function EthSwt_GetSwitchPortWakeupReason

Upstream requirements: SRS_Eth_00107

⌈

Service Name	EthSwt_GetSwitchPortWakeupReason	
Syntax	`Std_ReturnType EthSwt_GetSwitchPortWakeupReason ( uint8 SwitchIdx, uint8 SwitchPortIdx, EthTrcv_WakeupReasonType* Reason )`	
Service ID [hex]	0x4b	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	SwitchIdx	Index of the Ethernet switch within the context of the Ethernet Switch driver
	SwitchPortIdx	Index of the Ethernet switch port index in the context of the Ethernet switch driver
Parameters (inout)	None	
Parameters (out)	Reason	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 the indexed Ethernet switch port by calling EthTrcv_GetBusWuReason(...) of the referenced EthTrcv	
Available via	EthSwt.h	

⌋

[SWS_EthSwt_00442]

Upstream requirements: SRS_Eth_00107

⌈The function `EthSwt_GetSwitchPortWakeupReason` shall read the current wake-up reason of the indexed Ethernet switch port by forwarding the call to `EthTrcv_GetBusWuReason` of the referenced EthTrcv using the driver API abstraction of the EthIf. If the indexed Ethernet switch port has no reference to an EthTrcv, the function shall return `E_NOT_OK`.⌋

[SWS_EthSwt_00443]

Upstream requirements: SRS_BSW_00171

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

8.3.3.5 EthSwt_GetLinkState

[SWS_EthSwt_00037] Definition of API function EthSwt_GetLinkState

Upstream requirements: SRS_Eth_00119

⌈

Service Name	EthSwt_GetLinkState	
Syntax	`Std_ReturnType EthSwt_GetLinkState ( uint8 SwitchIdx, uint8 SwitchPortIdx, EthTrcv_LinkStateType* LinkStatePtr )`	
Service ID [hex]	0x06	
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	EthSwt.h	

⌋

[SWS_EthSwt_00038]

Upstream requirements: SRS_Eth_00118, SRS_Eth_00119

⌈The function `EthSwt_GetLinkState` shall read the current (link) state of the indexed switch port. If the indexed Ethernet port references an Ethernet transceiver, the link state shall be obtained by calling the function `EthTrcv_GetLinkState` of the Ethernet Transceiver Driver using the driver API abstraction of the EthIf. If the indexed Ethernet Switch port does not reference an Ethernet transceiver, the state shall be obtained from the MAC interface of the Switch port. If the MAC interface is not able to provide a link state (e.g. Ethernet hardware does not support a link state of the MAC interface), the API shall return the following state which is derived from the current mode:

- If the current mode of the indexed switch port is `ETH_MODE_ACTIVE`, then `ETHTRCV_LINK_STATE_ACTIVE` shall be returned

- If the current mode of the indexed switch port is ETH_MODE_DOWN, then ETHTRCV_LINK_STATE_DOWN shall be returned

⌋

[SWS_EthSwt_00042]

Upstream requirements: SRS_BSW_00171

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

8.3.3.6 EthSwt_GetBaudRate

[SWS_EthSwt_00044] Definition of API function EthSwt_GetBaudRate

Upstream requirements: SRS_Eth_00118

⌈

Service Name	EthSwt_GetBaudRate	
Syntax	`Std_ReturnType EthSwt_GetBaudRate ( uint8 SwitchIdx, uint8 SwitchPortIdx, EthTrcv_BaudRateType* BaudRatePtr )`	
Service ID [hex]	0x07	
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	EthSwt.h	

⌋

[SWS_EthSwt_00045]

Upstream requirements: SRS_Eth_00118

⌈The function EthSwt_GetBaudRate shall read the current baud rate of the indexed switch port. If the indexed Ethernet port reference an Ethernet transceiver, the baud rate shall be obtained by the function EthTrcv_GetBaudRate of the Ethernet Transceiver Driver using the driver API abstraction of the EthIf. If the indexed Ethernet Switch port does not reference an Ethernet transceiver, the baud rate shall be obtained from the MAC interface of the Switch port.⌋

[SWS_EthSwt_00049]

Upstream requirements: SRS_BSW_00171

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

8.3.3.7 EthSwt_GetDuplexMode

[SWS_EthSwt_00051] Definition of API function EthSwt_GetDuplexMode

Upstream requirements: SRS_Eth_00118

⌈

Service Name	EthSwt_GetDuplexMode	
Syntax	`Std_ReturnType EthSwt_GetDuplexMode ( uint8 SwitchIdx, uint8 SwitchPortIdx, EthTrcv_DuplexModeType* DuplexModePtr )`	
Service ID [hex]	0x08	
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	EthSwt.h	

⌋

[SWS_EthSwt_00052]

Upstream requirements: SRS_Eth_00118

⌈The function `EthSwt_GetDuplexMode` shall read the current duplex mode of the indexed switch port. If the indexed Ethernet port reference an Ethernet transceiver, the duplex mode shall be obtained by calling the function `EthTrcv_GetDuplexMode` of the Ethernet Transceiver Driver using the driver API abstraction of the EthIf. If the indexed Ethernet Switch port does not reference an Ethernet transceiver, the duplex mode shall be obtained from the MAC interface of the Switch port.⌋

[SWS_EthSwt_00056]

Upstream requirements: SRS_BSW_00171

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

8.3.3.8 EthSwt_GetMaxQueueBufferFillLevel

[SWS_EthSwt_91050] Definition of API function EthSwt_GetMaxQueueBufferFill Level ⌈

Service Name	EthSwt_GetMaxQueueBufferFillLevel	
Syntax	`Std_ReturnType EthSwt_GetMaxQueueBufferFillLevel ( uint8 SwitchIdx, uint8 SwitchPortIdx, uint8 SwitchPortEgressQueueIdx, uint32* SwitchPortEgressMaxQueueBufferFillLevelPtr )`	
Service ID [hex]	0x48	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different SwitchIdx and PortIdx. Non reentrant for the same SwitchIdx and Port Idx.	
Parameters (in)	SwitchIdx	Index of the Ethernet switch within the context of the Ethernet Switch Driver.
	SwitchPortIdx	Index of the Ethernet switch egress port at the addressed Ethernet switch.
	SwitchPortEgressQueue Idx	Index of the egress queue of the addressed Ethernet switch port
Parameters (inout)	None	
Parameters (out)	SwitchPortEgressMax QueueBufferFillLevelPtr	Pointer to a memory location, where the maximum amount of allocated queue buffer (in bytes) since the last read out shall be stored
Return value	Std_ReturnType	`E_OK`: The queue buffer fill level was written to the address pointed to by SwitchPortEgressMaxQueueBufferFillLevelPtr. `E_NOT_OK`: The maximal queue buffer level could not be obtained
Description	The function retrieves the maximum amount of allocated queue buffer of the indexed Ethernet switch egress port. If the Ethernet switch hardware does not support Ethernet switch port based maximal queue buffer level, the content of SwitchPortEgressMaxQueueBufferFillLevelPtr shall be set to 0xFFFFFFFF. This API may be called by e.g. a CDD.	
Available via	EthSwt.h	

⌋

[SWS_EthSwt_00430]

Upstream requirements: SRS_Eth_00119

⌈The function `EthSwt_GetMaxQueueBufferFillLevel` shall read out the maximum amount of allocated queue buffer since the last read out.⌋

[SWS_EthSwt_00431]

Upstream requirements: SRS_Eth_00119

⌈When the maximum amount of allocated queue buffer is read out, the value shall be reset to `0x00000000` explicitly, if it is not done by the hardware.⌋

[SWS_EthSwt_00432]

Upstream requirements: SRS_BSW_00171

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

8.3.3.9 EthSwt_GetPortIdentifier

[SWS_EthSwt_91015] Definition of API function EthSwt_GetPortIdentifier

Upstream requirements: SRS_Eth_00123

⌈

Service Name	EthSwt_GetPortIdentifier	
Syntax	`Std_ReturnType EthSwt_GetPortIdentifier ( uint8 SwitchIdx, uint8 PortIdx, uint32* OrgUniqueIdPtr, uint8* ModelNrPtr, uint8* RevisionNrPtr )`	
Service ID [hex]	0x34	
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)	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	EthSwt.h	

⌋

[SWS_EthSwt_00299]

Upstream requirements: SRS_Eth_00123

⌈The function `EthSwt_GetPortIdentifier` shall return the value of the organizationally unique identifier (OUI 24 bit) of the indexed Ethernet switch port that is connected to the indexed Ethernet switch. It shall set the 8 most significant bits of the OUI to 0xFFxxxxxx. If the Ethernet switch port references an Ethernet transceiver, the function shall obtain the OUI by calling the function `EthTrcv_GetPhyIdentifier` using the driver API abstraction of the EthIf and set the 8 most significant bits of the OUI to 0x00xxxxxx.⌋

[SWS_EthSwt_00394] ⌈If neither the Ethernet switch port nor the Ethernet Transceiver Driver can provide an OUI the function `EthSwt_GetPortIdentifier` shall return `E_NOT_OK`.⌋

[SWS_EthSwt_00303]

Upstream requirements: SRS_BSW_00171

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

8.3.3.10 EthSwt_PortEnableTimeStamp

[SWS_EthSwt_91028] Definition of API function EthSwt_PortEnableTimeStamp

Upstream requirements: SRS_Eth_00125

⌈

Service Name	EthSwt_PortEnableTimeStamp	
Syntax	`Std_ReturnType EthSwt_PortEnableTimeStamp ( uint8 CtrlIdx, Eth_BufIdxType BufIdx, EthSwt_MgmtInfoType* MgmtInfoPtr )`	
Service ID [hex]	0x40	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	CtrlIdx	Ethernet Controller index
	BufIdx	Ethernet Rx Buffer index
	MgmtInfoPtr	Management information including SwitchIdx and SwitchPortIdx
Parameters (inout)	None	
Parameters (out)	None	
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 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.	
Available via	EthSwt.h	

⌋

[SWS_EthSwt_00379]

Upstream requirements: SRS_BSW_00171

⌈The function `EthSwt_PortEnableTimeStamp` shall be pre compile time configurable ON/OFF by the configuration parameter: `EthSwtGlobalTimeSupportApi`.⌋

8.3.3.11 EthSwt_GetPortSignalQuality

[SWS_EthSwt_91014] Definition of API function EthSwt_GetPortSignalQuality

Upstream requirements: SRS_Eth_00123

⌈

Service Name	EthSwt_GetPortSignalQuality	
Syntax	`Std_ReturnType EthSwt_GetPortSignalQuality ( uint8 SwitchIdx, uint8 PortIdx, uint32* SignalQualityPtr )`	
Service ID [hex]	0x33	
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)	SignalQualityPtr	Pointer to the memory where the signal quality shall be stored.
Return value	Std_ReturnType	`E_OK`: signal quality could be read. `E_NOT_OK`: signal quality could not be read (i.e. no Ethernet transceiver is available for this Ethernet switch port)
Description	The function retrieves the signal quality of the link of the indexed Ethernet switch port. If no transceiver is referenced the signal quality shall be set to 0xFFFFFFFF.	
Available via	EthSwt.h	

⌋

[SWS_EthSwt_00293]

Upstream requirements: SRS_Eth_00123

⌈The function `EthSwt_GetPortSignalQuality` shall obtain the signal quality by calling the function `EthTrcv_GetPhySignalQuality` of the referenced Ethernet Transceiver Driver using the driver API abstraction of the EthIf. If the current signal quality is not available, the signal quality shall be set to `0xFFFFFFFF`.⌋

[SWS_EthSwt_00297]

Upstream requirements: SRS_BSW_00171

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

8.3.4 MAC address and VLAN related

8.3.4.1 EthSwt_GetPortMacAddr

[SWS_EthSwt_00060] Definition of API function EthSwt_GetPortMacAddr

Upstream requirements: SRS_Eth_00087, SRS_Eth_00181

⌈

Service Name	EthSwt_GetPortMacAddr	
Syntax	`Std_ReturnType EthSwt_GetPortMacAddr ( uint8 SwitchIdx, const uint8* MacAddrPtr, uint8* PortIdxPtr )`	
Service ID [hex]	0x09	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	SwitchIdx	Index of the switch within the context of the Ethernet Switch Driver
	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)	PortIdxPtr	Pointer to the port index
Return value	Std_ReturnType	`E_OK`: success `E_NOT_OK`: multiple ports were found
Description	Obtains the port over which this MAC-address at the indexed switch can be reached. The result might be used for a DHCP-server which will need the port/MAC-resolution. If for the PortIdxPtr the maximal possible value (255) is returned the given MAC address cannot be reached via a port of this switch. If multiple ports were found the API returns E_NOT_OK.	
Available via	EthSwt.h	

⌋

[SWS_EthSwt_00448] Behaviour if `EthSwtMacAddressLearningMode` is set to `IVL`

Upstream requirements: SRS_Eth_00087

⌈if `EthSwtMacAddressLearningMode` is set to `IVL` then the EthSwt driver shall return with `E_NOT_OK`.⌋

Note: If `EthSwtMacAddressLearningMode` is set to `IVL`, then the MAC-address to Ethernet switch port assignment is learned with respect to the according VLAN. With `IVL` the presence of an VLAN-ID is needed.

[SWS_EthSwt_00449] Behaviour if given MAC-address is available at exactly one Ethernet port

Upstream requirements: SRS_Eth_00087

⌈If the given MAC-address is available at exact one Ethernet switch port, then the EthSwt driver shall return the port index via PortIdxPtr and report `E_OK`.⌋

[SWS_EthSwt_00511] Behaviour if given MAC-address is available at multiple Ethernet ports

Upstream requirements: SRS_Eth_00087

⌈If the given MAC-address is available at multiple Ethernet switch ports, then the EthSwt driver shall return with `E_NOT_OK`.⌋

Note: If `EthSwtMacAddressLearningMode` is set to `SVL` and the MAC-address given with MacAddrPtr is a MAC multicast address, then the given MAC-address could be available at serveral Ethernet switch ports of the Ethernet switch addressed with SwitchIdx.

[SWS_EthSwt_00230]

Upstream requirements: SRS_BSW_00171

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

8.3.4.2 EthSwt_GetPortMacAddrVlan

[SWS_EthSwt_91051] Definition of API function EthSwt_GetPortMacAddrVlan

Upstream requirements: SRS_Eth_00087

⌈

Service Name	EthSwt_GetPortMacAddrVlan	
Syntax	`Std_ReturnType EthSwt_GetPortMacAddrVlan ( uint8 SwitchIdx, const uint8* MacAddrPtr, const uint16* VlanIdPtr, uint32* PortBitMapPtr )`	
Service ID [hex]	0x60	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	SwitchIdx	Index of the Ethernet switch within the context of the Ethernet Switch Driver
	MacAddrPtr	MAC-address which is requested to look-up the assignment to an Ethernet switch port
	VlanIdPtr	VlanId which is requested to look-up the assignment to an Ethernet switch port
Parameters (inout)	None	
Parameters (out)	PortBitMapPtr	Returns a pointer to an Ethernet switch port bit map, where the requested MAC-address with respect to the given VLAN-ID is available
Return value	Std_ReturnType	`E_OK`: success `E_NOT_OK`: request could not be successfully finalized, due to several possible reasons (e.g. requested Ethernet switch addressed with switchIdx is not valid or inactive)

▽

△

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

⌋

[SWS_EthSwt_00512] Behaviour if `EthSwtMacAddressLearningMode` is set to `SVL`

Upstream requirements: SRS_Eth_00087

⌈If `EthSwtMacAddressLearningMode` is set to `SVL`, then the EthSwt driver shall consider the given the MAC-address and ignore the given VLAN-ID to determine the MAC-address to Ethernet port assignment.⌋

Note: If `EthSwtMacAddressLearningMode` is set to `SVL`, the MAC-address to port assignment is shared across all VLANs and therefore the given VLAN is ignored.

[SWS_EthSwt_00513] Behaviour if `EthSwtMacAddressLearningMode` is set to `IVL` and VLAN-ID is set to value in a range from 1 to 4095

Upstream requirements: SRS_Eth_00087

⌈If `EthSwtMacAddressLearningMode` is set to `IVL` and given VLAN-ID is set to a value in a range from 1 to 4095, then the EthSwt driver shall consider the given the MAC-address and the given VLAN-ID to determine the MAC-address to Ethernet port assignment.⌋

[SWS_EthSwt_00514] Behaviour if `EthSwtMacAddressLearningMode` is set to `IVL` and VLAN-ID is set to 0

Upstream requirements: SRS_Eth_00087

⌈If `EthSwtMacAddressLearningMode` is set to `IVL` and given VLAN-ID is set to 0, then the EthSwt driver shall consider the given the MAC-address to determine the MAC-address to Ethernet port assignment irrespective of the VLAN.⌋

[SWS_EthSwt_00515] Behaviour if `EthSwtMacAddressLearningMode` is set to `IVL` and VLAN-ID is set to a value greater than 4095

Upstream requirements: SRS_Eth_00087

⌈If `EthSwtMacAddressLearningMode` is set to `IVL` and given VLAN-ID is set to a value greater that 4095, then the EthSwt driver shall return with `E_NOT_OK`.⌋

[SWS_EthSwt_00516] Creation of Ethernet switch port bit map

Upstream requirements: SRS_Eth_00087

⌈The function shall create a Ethernet switch port bit map addressed by PortBitMapPtr with respect to [SWS_EthSwt_00512], [SWS_EthSwt_00513] and [SWS_EthSwt_00514], where the index of the affected EthSwtPorts are encoded in a 32bit data type.

The least significant bit shall represent EthSwtPortIdx 0 and the most signification bit shall represent EthSwtPortIdx 31.⌋

[SWS_EthSwt_00517] Behaviour for creation of Ethernet switch port bit map

Upstream requirements: SRS_Eth_00087

⌈If the creation of the Ethernet switch port bit map was successful finalized and transfered to the location addressed with PortBitMapPtr, then the EthSwt driver shall return with `E_OK`.⌋

[SWS_EthSwt_00518] Behaviour if Ethernet switch port bit map exceeds uint32 data type

Upstream requirements: SRS_Eth_00087

⌈If the resulting Ethernet switch port bit map exceeds the unit32 data type, then the EthSwt driver shall return with `E_NOT_OK`.⌋

[SWS_EthSwt_00519] compile configuration for API

Upstream requirements: SRS_BSW_00171

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

8.3.4.3 EthSwt_GetArlTable

[SWS_EthSwt_00111] Definition of API function EthSwt_GetArlTable

Upstream requirements: SRS_Eth_00087, SRS_Eth_00181

⌈

Service Name	EthSwt_GetArlTable	
Syntax	`Std_ReturnType EthSwt_GetArlTable ( uint8 switchIdx, uint16* numberOfElements, Eth_MacVlanType* arlTableListPointer )`	
Service ID [hex]	0x0a	
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	EthSwt.h

⌋

[SWS_EthSwt_00228]

Upstream requirements: SRS_Eth_00087, SRS_Eth_00181

⌈The function `EthSwt_GetArlTable` shall provide a list of structs with MAC-address, VLAN-ID and port for the indexed switch.⌋

[SWS_EthSwt_00197]

Upstream requirements: SRS_Eth_00087, SRS_Eth_00181

⌈If the numberOfElements is greater 0x00, the arlTableListPointer shall be filled with up to numberOfElements elements. numberOfElements shall return the number of copied elements.⌋

[SWS_EthSwt_00235]

Upstream requirements: SRS_Eth_00087, SRS_Eth_00181

⌈The `EthSwt_GetArlTable` API shall return only the numberOfElements if the numberOfElements is set to 0x00. In this case no data will be copied and a NULLPTR can be used for the arlTableListPointer.⌋

[SWS_EthSwt_00229]

Upstream requirements: SRS_BSW_00171

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

8.3.4.4 EthSwt_EnableVlan

[SWS_EthSwt_00172] Definition of API function EthSwt_EnableVlan

Upstream requirements: SRS_Eth_00121, SRS_Eth_00114

⌈

Service Name	EthSwt_EnableVlan

Syntax	`Std_ReturnType EthSwt_EnableVlan (`
	`  uint8 SwitchIdx,`
	`  uint8 SwitchPortIdx,`
	`  uint16 VlanId,`
	`  boolean Enable`
	`)`

Service ID [hex]	0x12

▽

△

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
	VlanId	VLAN-ID to a preconfigured configuration on the given ingress port
	Enable	1 = VLAN-configuration enabled 0 = VLAN-configuration disabled (frames with given VLAN-ID will be dropped)
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: success `E_NOT_OK`: buffer level could not be obtained
Description	Enables or disables a pre-configured VLAN at a certain port of a switch.	
Available via	EthSwt.h	

⌋

[SWS_EthSwt_00173]

Upstream requirements: SRS_Eth_00121, SRS_Eth_00114

⌈The function `EthSwt_EnableVlan` shall enable or disable a pre-configured VLAN at a certain port of a switch.⌋

[SWS_EthSwt_00177]

Upstream requirements: SRS_BSW_00171

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

8.3.4.5 EthSwt_SetMacLearningMode

[SWS_EthSwt_00182] Definition of API function EthSwt_SetMacLearningMode

Upstream requirements: SRS_Eth_00087, SRS_Eth_00122

⌈

Service Name	EthSwt_SetMacLearningMode	
Syntax	`Std_ReturnType EthSwt_SetMacLearningMode ( uint8 SwitchIdx, uint8 SwitchPortIdx, EthSwt_MacLearningType MacLearningMode )`	
Service ID [hex]	0x15	
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

▽

△

	MacLearningMode	Defines whether MAC addresses shall be learned and if they shall be learned in software or hardware.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: success `E_NOT_OK`: configuration could be persistently reset
Description	Sets the MAC learning mode in one of the tree modes: 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	EthSwt.h	

⌋

[SWS_EthSwt_00183]

Upstream requirements: SRS_Eth_00122, SRS_Eth_00087

⌈The function `EthSwt_SetMacLearningMode` shall set the MAC learning mode according to `EthSwt_MacLearningType`.⌋

Note: This feature is hardware dependent, i.e. the switch hardware needs to support the different modes.

[SWS_EthSwt_00186]

Upstream requirements: SRS_BSW_00171

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

8.3.4.6 EthSwt_GetMacLearningMode

[SWS_EthSwt_00187] Definition of API function EthSwt_GetMacLearningMode

Upstream requirements: SRS_Eth_00087, SRS_Eth_00181

⌈

Service Name	EthSwt_GetMacLearningMode	
Syntax	`Std_ReturnType EthSwt_GetMacLearningMode ( uint8 SwitchIdx, uint8 SwitchPortIdx, EthSwt_MacLearningType* MacLearningMode )`	
Service ID [hex]	0x16	
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)	MacLearningMode	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	EthSwt.h	

⌋

[SWS_EthSwt_00188]

Upstream requirements: SRS_Eth_00087

⌈The function `EthSwt_GetMacLearningMode` shall return the MAC learning mode according to `EthSwt_MacLearningType`.⌋

Note: This feature is hardware dependent, i.e. the switch hardware needs to support the different learning modes.

[SWS_EthSwt_00191]

Upstream requirements: SRS_BSW_00171

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

8.3.5 MIB counter access

8.3.5.1 EthSwt_GetCounterValues

[SWS_EthSwt_00231] Definition of API function EthSwt_GetCounterValues

Upstream requirements: SRS_Eth_00128

⌈

Service Name	EthSwt_GetCounterValues	
Syntax	`Std_ReturnType EthSwt_GetCounterValues ( uint8 SwitchIdx, uint8 SwitchPortIdx, Eth_CounterType* CounterPtr )`	
Service ID [hex]	0x0c	
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	EthSwt.h

⌋

[SWS_EthSwt_00106]

Upstream requirements: SRS_Eth_00128

⌈ `EthSwt_GetCounterValues` shall read a list with drop counter values of the corresponding port of the switch. The meaning of these values is described at `Eth_CounterType`.⌋

8.3.5.2 EthSwt_GetRxStats

[SWS_EthSwt_00198] Definition of API function EthSwt_GetRxStats

Upstream requirements: SRS_Eth_00128

⌈

Service Name	EthSwt_GetRxStats	
Syntax	`Std_ReturnType EthSwt_GetRxStats ( uint8 SwitchIdx, uint8 SwitchPortIdx, Eth_RxStatsType* RxStats )`	
Service ID [hex]	0x0d	
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)	RxStats	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	EthSwt.h	

⌋

[SWS_EthSwt_00199]

Upstream requirements: SRS_Eth_00128

⌈`EthSwt_GetRxStats` shall return a list of statistic counters defined with `Eth_RxStatsType`. The majority of these Counters are derived from the IETF RFC2819.⌋

[SWS_EthSwt_00202]

Upstream requirements: SRS_BSW_00171

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

8.3.5.3 EthSwt_GetTxStats

[SWS_EthSwt_91001] Definition of API function EthSwt_GetTxStats

Upstream requirements: SRS_Eth_00128

⌈

Service Name	EthSwt_GetTxStats	
Syntax	`Std_ReturnType EthSwt_GetTxStats ( uint8 SwitchIdx, uint8 SwitchPortIdx, Eth_TxStatsType* TxStats )`	
Service ID [hex]	0x20	
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)	TxStats	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	Returns the list of Transmission Statistics out of IETF RFC1213 defined with Eth_TxStatsType, where the maximal possible value shall denote an invalid value, e.g. this counter is not available.	
Available via	EthSwt.h	

⌋

[SWS_EthSwt_00372]

Upstream requirements: SRS_Eth_00128

⌈`EthSwt_GetTxStats` shall return the list of Transmission Statistics out of IETF RFC1213 defined with `Eth_TxStatsType`, where the maximal possible value shall denote an invalid value, e.g. this counter is not available.⌋

[SWS_EthSwt_00362]

Upstream requirements: SRS_BSW_00171

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

8.3.5.4 EthSwt_GetTxErrorCounterValues

[SWS_EthSwt_91000] Definition of API function EthSwt_GetTxErrorCounterValues

Upstream requirements: SRS_Eth_00128

⌈

Service Name	EthSwt_GetTxErrorCounterValues	
Syntax	`Std_ReturnType EthSwt_GetTxErrorCounterValues ( uint8 SwitchIdx, uint8 SwitchPortIdx, Eth_TxErrorCounterValuesType* TxStats )`	
Service ID [hex]	0x21	
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)	TxStats	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	Returns the list of Transmission Error Counters out of IETF RFC1213 and RFC1643 defined with Eth_TxErrorCounterValuesType, where the maximal possible value shall denote an invalid value, e.g. this counter is not available.	
Available via	EthSwt.h	

⌋

[SWS_EthSwt_00373]

Upstream requirements: SRS_Eth_00128

⌈`EthSwt_GetTxErrorCounterValues` returns the list of Transmission Error Counters out of IETF RFC1213 and RFC1643 defined with `Eth_TxErrorCounterValuesType`, where the maximal possible value shall denote an invalid value, e.g. this counter is not available.⌋

[SWS_EthSwt_00370]

Upstream requirements: SRS_BSW_00171

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

8.3.6 Switch and transceiver management

8.3.6.1 EthSwt_GetSwitchReg

[SWS_EthSwt_00206] Definition of API function EthSwt_GetSwitchReg

Upstream requirements: SRS_Eth_00120

⌈

Service Name	EthSwt_GetSwitchReg	
Syntax	`Std_ReturnType EthSwt_GetSwitchReg ( uint8 SwitchIdx, uint32 page, uint32 register, uint32* registerContent )`	
Service ID [hex]	0x0e	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	SwitchIdx	Index of the switch within the context of the Ethernet Switch Driver
	page	Address of a register page
	register	Address of a register
Parameters (inout)	None	
Parameters (out)	registerContent	Content of the addresses register
Return value	Std_ReturnType	`E_OK`: success `E_NOT_OK`: drop counter could not be obtained
Description	Generic API for reading the content of a switch register	
Available via	EthSwt.h	

⌋

[SWS_EthSwt_00207]

Upstream requirements: SRS_Eth_00120

⌈The function `EthSwt_GetSwitchReg` shall read the content of a switch register.⌋

[SWS_EthSwt_00210]

Upstream requirements: SRS_BSW_00171

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

8.3.6.2 EthSwt_SetSwitchReg

[SWS_EthSwt_00211] Definition of API function EthSwt_SetSwitchReg

Upstream requirements: SRS_Eth_00120

⌈

Service Name	EthSwt_SetSwitchReg	
Syntax	`Std_ReturnType EthSwt_SetSwitchReg ( uint8 SwitchIdx, uint32 page, uint32 register, uint32 registerContent )`	
Service ID [hex]	0x0f	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	SwitchIdx	Index of the switch within the context of the Ethernet Switch Driver
	page	Address of a register page
	register	Address of a register
	registerContent	Content of the addresses register
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: success `E_NOT_OK`: drop counter could not be obtained
Description	Generic API for writing the content of a switch register	
Available via	EthSwt.h	

⌋

[SWS_EthSwt_00212]

Upstream requirements: SRS_Eth_00120

⌈The function `EthSwt_SetSwitchReg` shall write the content to the switch register.⌋

[SWS_EthSwt_00215]

Upstream requirements: SRS_BSW_00171

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

8.3.6.3 EthSwt_ReadTrcvRegister

[SWS_EthSwt_00216] Definition of API function EthSwt_ReadTrcvRegister

Upstream requirements: SRS_Eth_00120

⌈

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

⌋

[SWS_EthSwt_00217]

Upstream requirements: SRS_Eth_00118, SRS_Eth_00120

⌈The function `EthSwt_ReadTrcvRegister` shall read the specified transceiver register through the MII or SPI of the indexed switch port.⌋

[SWS_EthSwt_00220]

Upstream requirements: SRS_BSW_00171

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

8.3.6.4 EthSwt_WriteTrcvRegister

[SWS_EthSwt_00221] Definition of API function EthSwt_WriteTrcvRegister

Upstream requirements: SRS_Eth_00120

⌈

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

⌋

[SWS_EthSwt_00222]

Upstream requirements: SRS_Eth_00118, SRS_Eth_00120

⌈The function `EthSwt_WriteTrcvRegister` shall write the specified transceiver register through the MII or SPI of the indexed switch port.⌋

[SWS_EthSwt_00225]

Upstream requirements: SRS_BSW_00171

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

8.3.6.5 EthSwt_ReadMmd

[SWS_EthSwt_91052] Definition of API function EthSwt_ReadMmd ⌈

Service Name	EthSwt_ReadMmd	
Syntax	`Std_ReturnType EthSwt_ReadMmd ( uint8 SwitchIdx, uint8 SwitchPortIdx, uint8 Mmd, uint16 RegIdx, uint16* RegValPtr )`	
Service ID [hex]	0xa6	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different SwitchIdx, non reentrant for same SwitchIdx	
Parameters (in)	SwitchIdx	Index of the switch.
	SwitchPortIdx	Index of the port at the addressed switch
	Mmd	MDIO Manageable Device
	RegIdx	Index of the transceiver register on the SMI.
Parameters (inout)	None	
Parameters (out)	RegValPtr	Filled with the register content of the indexed register
Return value	Std_ReturnType	`E_OK`: Service accepted `E_NOT_OK`: Service denied
Description	Reads a transceiver register using Clause 45 access if supported by hardware or implements a Clause 45 access using Clause 22 operations.	
Available via	EthSwt.h	

⌋

[SWS_EthSwt_00560] EthSwt_ReadMmd functionality ⌈The function `EthSwt_ReadMmd` shall read the specified transceiver register through the MII or SPI of the indexed switch port.⌋

[SWS_EthSwt_00561] Configuring EthSwt_ReadMmd availability ⌈The function `EthSwt_ReadMmd` shall be precompile time configurable On/Off by the configuration parameter: `EthSwtReadMmdApi`.⌋

8.3.6.6 EthSwt_WriteMmd

[SWS_EthSwt_91053] Definition of API function EthSwt_WriteMmd ⌈

Service Name	EthSwt_WriteMmd
Syntax	`Std_ReturnType EthSwt_WriteMmd ( uint8 SwitchIdx, uint8 SwitchPortIdx, uint8 Mmd, uint16 RegIdx, uint16 RegVal )`
Service ID [hex]	0xa7

▽

△

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

⌋

[SWS_EthSwt_00562] EthSwt_WriteMmd functionality ⌈The function `EthSwt_WriteMmd` shall write a value to the specified transceiver register through the MII of the indexed switch port.⌋

[SWS_EthSwt_00563] Configuring EthSwt_WriteMmd availability ⌈The function `EthSwt_WriteMmd` shall be precompile time configurable On/Off by the configuration parameter: `EthSwtWriteMmdApi`.⌋

8.3.7 Frame Handling

8.3.7.1 EthSwt_EthRxProcessFrame

[SWS_EthSwt_91004] Definition of API function EthSwt_EthRxProcessFrame

Upstream requirements: SRS_Eth_00125

⌈

Service Name	EthSwt_EthRxProcessFrame	

Syntax	`Std_ReturnType EthSwt_EthRxProcessFrame (`
	`  uint8 CtrlIdx,`
	`  Eth_BufIdxType BufIdx,`
	`  uint8** DataPtr,`
	`  uint16* LengthPtr,`
	`  boolean* IsMgmtFrameOnlyPtr`
	`)`	

Service ID [hex]	0xb0	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different CtrlIdxs. Non reentrant for the same CtrlIdx.	
Parameters (in)	CtrlIdx	Ethernet Controller index
	BufIdx	Ethernet Rx Buffer index

▽

△

Parameters (inout)	DataPtr	IN: Pointer to the position of the EtherType of a common Ethernet frame OUT: Pointer to the position of the EtherType in the management frame
	LengthPtr	IN: Pointer to the length of the frame received OUT: Pointer to the length decreased by the management information length.
Parameters (out)	IsMgmtFrameOnlyPtr	Information about the kind of frame FALSE: Frame is not only for management purpose, but also for normal communication. TRUE: Frame is only for management purpose and must not be processed in common receive process
Return value	Std_ReturnType	`E_OK`: Frame successfully processed `E_NOT_OK`: Frame processing failed
Description	Function inspects the Ethernet frame passed by the data pointer for management information and stores it for later use in the corresponding EthRxFinishedIndication call.	
Available via	EthSwt_Eth.h	

⌋

[SWS_EthSwt_00249]

Upstream requirements: SRS_BSW_00171

⌈The function `EthSwt_EthRxProcessFrame` shall be pre compile time configurable ON/OFF by the configuration parameter: `EthSwtManagementSupportApi` .⌋

8.3.7.2 EthSwt_EthRxFinishedIndication

[SWS_EthSwt_91005] Definition of API function EthSwt_EthRxFinishedIndication

Upstream requirements: SRS_Eth_00125

⌈

Service Name	EthSwt_EthRxFinishedIndication	
Syntax	`Std_ReturnType EthSwt_EthRxFinishedIndication ( uint8 CtrlIdx, Eth_BufIdxType BufIdx )`	
Service ID [hex]	0xb1	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different CtrlIdxs. Non reentrant for the same CtrlIdx.	
Parameters (in)	CtrlIdx	Ethernet Controller index
	BufIdx	Ethernet Rx Buffer index
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: Frame successfully processed `E_NOT_OK`: Frame processing failed

▽

△

Description	Function inspects the Ethernet frame passed by the data pointer for management information and stores it for later use in EthSwt_EthRxFinishedIndication().
Available via	EthSwt_Eth.h

⌋

[SWS_EthSwt_00253]

Upstream requirements: SRS_BSW_00171

⌈The function `EthSwt_EthRxFinishedIndication` shall be pre compile time configurable ON/OFF by the configuration parameter: `EthSwtManagementSupportApi`.⌋

8.3.7.3 EthSwt_EthTxPrepareFrame

[SWS_EthSwt_91006] Definition of API function EthSwt_EthTxPrepareFrame

Upstream requirements: SRS_Eth_00125

⌈

Service Name	EthSwt_EthTxPrepareFrame	
Syntax	`Std_ReturnType EthSwt_EthTxPrepareFrame ( uint8 CtrlIdx, Eth_BufIdxType BufIdx, uint8** DataPtr, uint16* LengthPtr )`	
Service ID [hex]	0xb5	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different CtrlIdxs. Non reentrant for the same CtrlIdx.	
Parameters (in)	CtrlIdx	Ethernet Controller index
	BufIdx	Ethernet Rx Buffer index
Parameters (inout)	DataPtr	IN: Pointer to the position of the EtherType of a common Ethernet frame OUT: Pointer to the position of the EtherType in the management frame
	LengthPtr	IN: Pointer to the length of the buffer without management information OUT: Pointer to the modified length needed for buffer and management information
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: Frame successfully prepared `E_NOT_OK`: Frame preparation failed
Description	Function inspects the Ethernet frame passed by the data pointer for management information and stores it for later use in EthSwt_EthRxFinishedIndication().	
Available via	EthSwt_Eth.h	

⌋

[SWS_EthSwt_00257]

Upstream requirements: SRS_BSW_00171

⌈The function `EthSwt_EthTxPrepareFrame` shall be pre compile time configurable ON/OFF by the configuration parameter: `EthSwtManagementSupportApi` .⌋

8.3.7.4 EthSwt_EthTxAdaptBufferLength

[SWS_EthSwt_91007] Definition of API function EthSwt_EthTxAdaptBuffer Length

Upstream requirements: SRS_Eth_00125

⌈

Service Name	EthSwt_EthTxAdaptBufferLength	
Syntax	`void EthSwt_EthTxAdaptBufferLength ( uint16* LengthPtr )`	
Service ID [hex]	0xb2	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	None	
Parameters (inout)	LengthPtr	IN: Pointer to the length of the buffer without management information. OUT: Pointer to the modified length needed for buffer and management information.
Parameters (out)	None	
Return value	None	
Description	Modifies the buffer length to be able to insert management information.	
Available via	EthSwt_Eth.h	

⌋

[SWS_EthSwt_00261]

Upstream requirements: SRS_BSW_00171

⌈The function `EthSwt_EthTxAdaptBufferLength` shall be pre compile time configurable ON/OFF by the configuration parameter: `EthSwtManagementSupportApi` .⌋

8.3.7.5 EthSwt_EthTxProcessFrame

[SWS_EthSwt_91009] Definition of API function EthSwt_EthTxProcessFrame

Upstream requirements: SRS_Eth_00125

⌈

Service Name	EthSwt_EthTxProcessFrame	
Syntax	`Std_ReturnType EthSwt_EthTxProcessFrame ( uint8 CtrlIdx, Eth_BufIdxType BufIdx, uint8** DataPtr, uint16* LengthPtr )`	
Service ID [hex]	0xb4	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different CtrlIdxs. Non reentrant for the same CtrlIdx.	
Parameters (in)	CtrlIdx	Ethernet Controller index
	BufIdx	Ethernet Tx Buffer index
Parameters (inout)	DataPtr	IN: Pointer to the position of the EtherType of a common Ethernet frame OUT: Pointer to the position of the EtherType in the management frame
	LengthPtr	IN: Pointer to the length of the received frame OUT: Pointer to the length decreased by the management information length
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: Frame successfully processed `E_NOT_OK`: Frame processing failed
Description	Function inspects the Ethernet frame passed by the data pointer for management information and stores it for later use in EthSwt_EthRxFinishedIndication().	
Available via	EthSwt_Eth.h	

⌋

[SWS_EthSwt_00268]

Upstream requirements: SRS_BSW_00171

⌈The function `EthSwt_EthTxProcessFrame` shall be pre compile time configurable ON/OFF by the configuration parameter: `EthSwtManagementSupportApi` .⌋

8.3.7.6 EthSwt_EthTxFinishedIndication

[SWS_EthSwt_91010] Definition of API function EthSwt_EthTxFinishedIndication

Upstream requirements: SRS_Eth_00125

⌈

Service Name	EthSwt_EthTxFinishedIndication	
Syntax	`Std_ReturnType EthSwt_EthTxFinishedIndication ( uint8 CtrlIdx, Eth_BufIdxType BufIdx )`	
Service ID [hex]	0xb6	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different CtrlIdxs. Non reentrant for the same CtrlIdx.	
Parameters (in)	CtrlIdx	Ethernet Controller index
	BufIdx	Ethernet Rx Buffer index
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: Frame successfully processed `E_NOT_OK`: Frame processing failed
Description	Function inspects the Ethernet frame passed by the data pointer for management information and stores it for later use in EthSwt_EthRxFinishedIndication().	
Available via	EthSwt_Eth.h	

⌋

[SWS_EthSwt_00273]

Upstream requirements: SRS_BSW_00171

⌈The function `EthSwt_EthTxFinishedIndication` shall be pre compile time configurable ON/OFF by the configuration parameter: `EthSwtManagementSupportApi`.⌋

8.3.8 Port Mirroring

8.3.8.1 EthSwt_WritePortMirrorConfiguration

[SWS_EthSwt_91018] Definition of API function EthSwt_WritePortMirrorConfiguration

Upstream requirements: SRS_Eth_00123

⌈

Service Name	EthSwt_WritePortMirrorConfiguration
Syntax	`Std_ReturnType EthSwt_WritePortMirrorConfiguration ( uint8 MirroredSwitchIdx, const EthSwt_PortMirrorCfgType* PortMirrorConfigurationPtr )`
Service ID [hex]	0x36

▽

△

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	Pointer of the port configuration, which shall be stored in a shadow buffer in the Ethernet switch driver
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	EthSwt.h	

⌋

[SWS_EthSwt_00309]

Upstream requirements: SRS_Eth_00123

⌈The function EthSwt_WritePortMirrorConfiguration shall store the port mirror configuration of the given MirroredSwitchIdx in a shadow buffer. The MirroredSwitchIdx shall be used to identify the port mirror configuration within the Ethernet switch driver.⌋

[SWS_EthSwt_00312]

Upstream requirements: SRS_BSW_00171

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

[SWS_EthSwt_00424]

Upstream requirements: SRS_Eth_00123

⌈The function shall return with ETHSWT_PORT_MIRRORING_CONFIGURATION_NOT_SUPPORTED, if the port mirroring configuration is not supported by the Ethernet switch driver or by the Ethernet switch hardware, e.g.:

- the configured mirrored traffic direction (see [SWS_EthSwt_91017] "TrafficDirectionIngressBitMask" and "TrafficDirectionEgressBitMask") for ingress and egress traffic of the same port is not supported
- mirrored ports and capture ports, respectively, are not available within the Ethernet switch driver

⌋

8.3.8.2 EthSwt_ReadPortMirrorConfiguration

[SWS_EthSwt_91019] Definition of API function EthSwt_ReadPortMirrorConfiguration

Upstream requirements: SRS_Eth_00123

⌈

Service Name	EthSwt_ReadPortMirrorConfiguration	
Syntax	`Std_ReturnType EthSwt_ReadPortMirrorConfiguration ( uint8 MirroredSwitchIdx, EthSwt_PortMirrorCfgType* PortMirrorConfigurationPtr )`	
Service ID [hex]	0x37	
Sync/Async	Asynchronous	
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 read successfully. (i.e. indexed Ethernet switch is not available)
Description	Obtain the port mirror configuration of the given Ethernet switch.	
Available via	EthSwt.h	

⌋

[SWS_EthSwt_00313]

Upstream requirements: SRS_Eth_00123

⌈The function EthSwt_ReadPortMirrorConfiguration shall return the port mirror configuration identified by the given MirroredSwitchIdx. If no port mirror configuration is found for the MirroredSwitchIdx, the function shall return E_NOT_OK.⌋

[SWS_EthSwt_00317]

Upstream requirements: SRS_BSW_00171

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

8.3.8.3 EthSwt_DeletePortMirrorConfiguration

[SWS_EthSwt_91034] Definition of API function EthSwt_DeletePortMirrorConfiguration ⌈

Service Name	EthSwt_DeletePortMirrorConfiguration	
Syntax	`Std_ReturnType EthSwt_DeletePortMirrorConfiguration ( uint8 MirroredSwitchIdx )`	
Service ID [hex]	0x4a	
Sync/Async	Synchronous	
Reentrancy	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	EthSwt.h	

⌋

[SWS_EthSwt_00425]

Upstream requirements: SRS_Eth_00123

⌈The function `EthSwt_DeletePortMirrorConfiguration` shall mark the stored port mirror configuration in the shadow buffer of the given MirroredSwitchIdx as "to be deleted".⌋

[SWS_EthSwt_00426]

Upstream requirements: SRS_Eth_00123

⌈If a port mirroring for the given MirroredSwitchIdx is enabled, the request to delete the configuration shall be rejected by returning `E_NOT_OK`. Only those port configurations are allowed to be deleted, where the port mirroring of the given MirroredSwitchIdx is disabled.⌋

[SWS_EthSwt_00427]

Upstream requirements: SRS_BSW_00171

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

8.3.8.4 EthSwt_GetPortMirrorState

[SWS_EthSwt_91021] Definition of API function EthSwt_GetPortMirrorState

Upstream requirements: SRS_Eth_00123

⌈

Service Name	EthSwt_GetPortMirrorState	
Syntax	`Std_ReturnType EthSwt_GetPortMirrorState ( uint8 SwitchIdx, uint8 PortIdx, EthSwt_PortMirrorStateType* PortMirrorStatePtr )`	
Service ID [hex]	0x38	
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	EthSwt.h	

⌋

[SWS_EthSwt_00318]

Upstream requirements: SRS_Eth_00123

⌈The function `EthSwt_GetPortMirrorState` shall return the port mirroring state of the indexed ethernet switch port.⌋

[SWS_EthSwt_00322]

Upstream requirements: SRS_BSW_00171

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

8.3.8.5 EthSwt_SetPortMirrorState

[SWS_EthSwt_91022] Definition of API function EthSwt_SetPortMirrorState

Upstream requirements: SRS_Eth_00123

⌈

Service Name	EthSwt_SetPortMirrorState	
Syntax	`Std_ReturnType EthSwt_SetPortMirrorState ( uint8 MirroredSwitchIdx, EthSwt_PortMirrorStateType PortMirrorState )`	
Service ID [hex]	0x39	
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.
	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	`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	EthSwt.h	

⌋

[SWS_EthSwt_00323]

Upstream requirements: SRS_Eth_00123

⌈The function `EthSwt_SetPortMirrorState` shall request the given port mirroring state for the port mirroring configuration of the indexed Ethernet switch, and store the requested port mirror state in a shadow buffer.⌋

[SWS_EthSwt_00327]

Upstream requirements: SRS_BSW_00171

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

8.3.9 Test modes and cable diagnostic

8.3.9.1 EthSwt_SetPortTestMode

[SWS_EthSwt_91029] Definition of API function EthSwt_SetPortTestMode

Upstream requirements: SRS_Eth_00123

⌈

Service Name	EthSwt_SetPortTestMode	
Syntax	`Std_ReturnType EthSwt_SetPortTestMode ( uint8 SwitchIdx, uint8 PortIdx, EthTrcv_PhyTestModeType Mode )`	
Service ID [hex]	0x3a	
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	EthSwt.h	

⌋

[SWS_EthSwt_00328]

Upstream requirements: SRS_Eth_00123

⌈The function `EthSwt_SetPortTestMode` shall forward the call with the given test mode by calling the function `EthTrcv_SetPhyTestMode` of the referenced Ethernet Transceiver Driver using the driver API abstraction of the EthIf.⌋

[SWS_EthSwt_00332]

Upstream requirements: SRS_BSW_00171

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

8.3.9.2 EthSwt_SetPortLoopbackMode

[SWS_EthSwt_91023] Definition of API function EthSwt_SetPortLoopbackMode

Upstream requirements: SRS_Eth_00123

⌈

Service Name	EthSwt_SetPortLoopbackMode	
Syntax	`Std_ReturnType EthSwt_SetPortLoopbackMode ( uint8 SwitchIdx, uint8 PortIdx, EthTrcv_PhyLoopbackModeType Mode )`	
Service ID [hex]	0x3b	
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 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	EthSwt.h	

⌋

[SWS_EthSwt_00334]

Upstream requirements: SRS_Eth_00123

⌈The function `EthSwt_SetPortLoopbackMode` shall forward the call with the given loop-back mode by calling the function `EthTrcv_SetPhyLoopbackMode` of the referenced Ethernet Transceiver Driver using the driver API abstraction of the EthIf.⌋

[SWS_EthSwt_00338]

Upstream requirements: SRS_BSW_00171

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

8.3.9.3 EthSwt_RunPortCableDiagnostic

[SWS_EthSwt_91011] Definition of API function EthSwt_RunPortCableDiagnostic ⌈

Service Name	EthSwt_RunPortCableDiagnostic	
Syntax	`Std_ReturnType EthSwt_RunPortCableDiagnostic ( uint8 SwitchIdxIdx, uint8 PortIdx )`	
Service ID [hex]	0x45	
Sync/Async	Asynchronous	
Reentrancy	Reentrant for different SwitchIdx and PortIdx. Non reentrant for the same SwitchIdx and PortIdx.	
Parameters (in)	SwitchIdxIdx	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	EthSwt.h	

⌋

[SWS_EthSwt_00429] ⌈The function `EthSwt_RunPortCableDiagnostic` shall forward the call by calling `EthTrcv_RunCableDiagnostic` of the referenced Ethernet Transceiver Driver using the driver API abstraction of the EthIf.⌋

8.3.9.4 EthSwt_GetPortCableDiagnosticsResult

[SWS_EthSwt_91025] Definition of API function EthSwt_GetPortCableDiagnosticsResult

Upstream requirements: SRS_Eth_00123

⌈

Service Name	EthSwt_GetPortCableDiagnosticsResult	
Syntax	`Std_ReturnType EthSwt_GetPortCableDiagnosticsResult ( uint8 SwitchIdx, uint8 PortIdx, EthTrcv_CableDiagResultType* ResultPtr )`	
Service ID [hex]	0x3f	
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	EthSwt.h	

⌋

[SWS_EthSwt_00346]

Upstream requirements: SRS_Eth_00123

⌈The function `EthSwt_GetPortCableDiagnosticsResult` shall obtain the cable diagnostics result by calling the function `EthTrcv_GetCableDiagnosticsResult` of the referenced Ethernet Transceiver Driver using the driver API abstraction of the EthIf.⌋

[SWS_EthSwt_00350]

Upstream requirements: SRS_BSW_00171

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

8.3.9.5 EthSwt_SetPortTxMode

[SWS_EthSwt_91024] Definition of API function EthSwt_SetPortTxMode

Upstream requirements: SRS_Eth_00123

⌈

Service Name	EthSwt_SetPortTxMode	
Syntax	`Std_ReturnType EthSwt_SetPortTxMode ( uint8 SwitchIdx, uint8 PortIdx, EthTrcv_PhyTxModeType Mode )`	
Service ID [hex]	0x3c	
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	EthSwt.h	

⌋

[SWS_EthSwt_00340]

Upstream requirements: SRS_Eth_00123

⌈The function `EthSwt_SetPortTxMode` shall forward the call with the given transmission mode by calling the function `EthTrcv_SetPhyTxMode` of the referenced Ethernet Transceiver Driver using the driver API abstraction of the EthIf.⌋

[SWS_EthSwt_00344]

Upstream requirements: SRS_BSW_00171

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

8.3.9.6 EthSwt_PortLinkStateRequest

[SWS_EthSwt_91123] Definition of API function EthSwt_PortLinkStateRequest ⌈

Service Name	EthSwt_PortLinkStateRequest	
Syntax	`Std_ReturnType EthSwt_PortLinkStateRequest ( uint8 SwitchIdx, uint8 PortIdx, EthTrcv_LinkStateType PortLinkState )`	
Service ID [hex]	0x49	
Sync/Async	Asynchronous	
Reentrancy	Reentrant for different SwitchIdx and PortIdx. Non reentrant for the same SwitchIdx and Port Idx.	
Parameters (in)	SwitchIdx	Index of the switch within the context of the Ethernet Switch Driver.
	PortIdx	Index of the port at the addressed switch.
	PortLinkState	The Ethernet link state of a physical Ethernet connection.
Parameters (inout)	None	
Parameters (out)	None	

▽

△

Return value	Std_ReturnType	E_OK: Request has been accepted and if the function call is in state ETHSWT_STATE_PORTINIT_COMPLETED or ETHSWT_STATE_ACTIVE E_NOT_OK: Request has not been accepted. (e.g. the indexed Ethernet switch port does not reference an EthTrcv)
Description	Request a link state by calling EthTrcv_TransceiverLinkStateRequest with the TrcvIdx of the Ethernet transceiver which is referenced by the Ethernet Switch port (PortIdx).	
Available via	EthSwt.h	

⌋

[SWS_EthSwt_00415] ⌈The function EthSwt_PortLinkStateRequest shall request the given link state for the indexed Ethernet switch port of the switch by calling the EthTrcv_TransceiverLinkStateRequest with the given EthTrcv_LinkStateType using the driver API abstraction of the EthIf. If the EthSwtPort does not reference an EthTrcv, then the function shall return E_NOT_OK.⌋

8.3.10 Switch Management

8.3.10.1 EthSwt_GetRxMgmtObject

[SWS_EthSwt_91038] Definition of API function EthSwt_GetRxMgmtObject ⌈

Service Name	EthSwt_GetRxMgmtObject	
Syntax	`Std_ReturnType EthSwt_GetRxMgmtObject ( uint8 CtrlIdx, const uint8* 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	Pointer to the management object.
Return value	Std_ReturnType	E_OK: success E_NOT_OK: management object could not be obtained
Description	Obtains the MgmtObject of the (in this context) unique DataPtr.	
Available via	EthSwt.h	

⌋

8.3.10.2 EthSwt_GetTxMgmtObject

[SWS_EthSwt_91039] Definition of API function EthSwt_GetTxMgmtObject ⌈

Service Name	EthSwt_GetTxMgmtObject	
Syntax	`Std_ReturnType EthSwt_GetTxMgmtObject ( uint8 CtrlIdx, Eth_BufIdxType BufIdx, EthSwt_MgmtObjectType** MgmtObjectPtr )`	
Service ID [hex]	0x44	
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	Obtains the MgmtObject of the (in this context) unique BufIdx.	
Available via	EthSwt.h	

⌋

8.3.10.3 EthSwt_SetMgmtInfo

[SWS_EthSwt_91008] Definition of API function EthSwt_SetMgmtInfo

Upstream requirements: SRS_Eth_00125

⌈

Service Name	EthSwt_SetMgmtInfo	
Syntax	`Std_ReturnType EthSwt_SetMgmtInfo ( uint8 CtrlIdx, Eth_BufIdxType BufIdx, const EthSwt_MgmtInfoType* MgmtInfoPtr )`	
Service ID [hex]	0x27	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	CtrlIdx	Ethernet Controller index
	BufIdx	Ethernet Rx 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	Extends the Ethernet frame prepared previously by EthSwt_EthTxPrepareFrame() with the management information to achieve transmission only on specific ports.	
Available via	EthSwt.h	

⌋

[SWS_EthSwt_00264]

Upstream requirements: SRS_BSW_00171

⌈The function `EthSwt_SetMgmtInfo` shall be pre compile time configurable ON/OFF by the configuration parameter: `EthSwtManagementSupportApi` .⌋

8.3.11 MacSec API

8.3.11.1 EthSwt_MacSecUpdateSecY

[SWS_EthSwt_91124] Definition of API function EthSwt_MacSecUpdateSecY

Status: DRAFT

⌈

Service Name	EthSwt_MacSecUpdateSecY (draft)	

Syntax	`Std_ReturnType EthSwt_MacSecUpdateSecY (`
	`  const EthSwt_MgmtInfoType* MgmtInfoPtr,`
	`  const Mka_MacSecConfigType* MACsecCfgPtr,`
	`  uint64 TxSci`
	`)`	

Service ID [hex]	0x4d	
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 EthSwt (EthSwt Config/EthSwtIdx), PortIdx in context of EthSwt (EthSwtPort/Eth SwtPortIdx).
	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 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	EthSwt.h	

⌋

8.3.11.2 EthSwt_MacSecInitRxSc

[SWS_EthSwt_91125] Definition of API function EthSwt_MacSecInitRxSc

Status: DRAFT

⌈

Service Name	EthSwt_MacSecInitRxSc (draft)	

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

Service ID [hex]	0x4e	
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 EthSwt (EthSwt Config/EthSwtIdx), PortIdx in context of EthSwt (EthSwtPort/Eth SwtPortIdx).
	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	EthSwt.h	

⌋

8.3.11.3 EthSwt_MacSecResetRxSc

[SWS_EthSwt_91126] Definition of API function EthSwt_MacSecResetRxSc

Status: DRAFT

⌈

Service Name	EthSwt_MacSecResetRxSc (draft)	

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

Service ID [hex]	0x4f	
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 EthSwt (EthSwt Config/EthSwtIdx), PortIdx in context of EthSwt (EthSwtPort/Eth SwtPortIdx).

▽

△

	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 delete the Reception Secure Channel addressed with the given Reception Secure Channel Identifier. **Tags:** atp.Status=draft	
Available via	EthSwt.h	

⌋

8.3.11.4 EthSwt_MacSecAddTxSa

[SWS_EthSwt_91127] Definition of API function EthSwt_MacSecAddTxSa

Status: DRAFT

⌈

Service Name	EthSwt_MacSecAddTxSa (draft)	

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

Service ID [hex]	0x50	
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 EthSwt (EthSwt Config/EthSwtIdx), PortIdx in context of EthSwt (EthSwtPort/Eth SwtPortIdx).
	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 Transceiver. The Short Secure Channel Identifier is included to support XPN configurations. **Tags:** atp.Status=draft
Available via	EthSwt.h

⌋

8.3.11.5 EthSwt_MacSecUpdateTxSa

[SWS_EthSwt_91128] Definition of API function EthSwt_MacSecUpdateTxSa

Status: DRAFT

⌈

Service Name	EthSwt_MacSecUpdateTxSa (draft)	

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

Service ID [hex]	0x51	
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 EthSwt (EthSwt Config/EthSwtIdx), PortIdx in context of EthSwt (EthSwtPort/Eth SwtPortIdx).
	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	EthSwt.h	

⌋

8.3.11.6 EthSwt_MacSecDeleteTxSa

[SWS_EthSwt_91129] Definition of API function EthSwt_MacSecDeleteTxSa

Status: DRAFT

⌈

Service Name	EthSwt_MacSecDeleteTxSa (draft)	
Syntax	`Std_ReturnType EthSwt_MacSecDeleteTxSa ( const EthSwt_MgmtInfoType* MgmtInfoPtr, uint8 An )`	
Service ID [hex]	0x52	
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 EthSwt (EthSwt Config/EthSwtIdx), PortIdx in context of EthSwt (EthSwtPort/EthSwtPortIdx).
	An	Association Number to use in the MACsec's transmission secure association
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: The request has been accepted `E_NOT_OK`: The request has not been accepted
Description	Request the Ethernet Switch Driver to remove the Transmission Secure Association identified by the provided Association Number. **Tags:** atp.Status=draft	
Available via	EthSwt.h	

⌋

8.3.11.7 EthSwt_MacSecAddRxSa

[SWS_EthSwt_91130] Definition of API function EthSwt_MacSecAddRxSa

Status: DRAFT

⌈

Service Name	EthSwt_MacSecAddRxSa (draft)
Syntax	`Std_ReturnType EthSwt_MacSecAddRxSa ( const EthSwt_MgmtInfoType* MgmtInfoPtr, uint8 An, uint64 LowestPn, uint32 Ssci, const Mka_SakKeyPtrType* KeysPtr, boolean Active )`
Service ID [hex]	0x53
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 EthSwt (EthSwt Config/EthSwtIdx), PortIdx in context of EthSwt (EthSwtPort/Eth SwtPortIdx).
	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 Transceiver. The Short Secure Channel Identifier is included to support XPN configurations. **Tags:** atp.Status=draft	
Available via	EthSwt.h	

⌋

8.3.11.8 EthSwt_MacSecUpdateRxSa

[SWS_EthSwt_91131] Definition of API function EthSwt_MacSecUpdateRxSa

Status: DRAFT

⌈

Service Name	EthSwt_MacSecUpdateRxSa (draft)	
Syntax	`Std_ReturnType EthSwt_MacSecUpdateRxSa ( const EthSwt_MgmtInfoType* MgmtInfoPtr, uint8 An, uint64 LowestPn, boolean Active )`	
Service ID [hex]	0x54	
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 EthSwt (EthSwt Config/EthSwtIdx), PortIdx in context of EthSwt (EthSwtPort/Eth SwtPortIdx).
	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	EthSwt.h	

⌋

8.3.11.9 EthSwt_MacSecDeleteRxSa

[SWS_EthSwt_91132] Definition of API function EthSwt_MacSecDeleteRxSa

Status: DRAFT

⌈

Service Name	EthSwt_MacSecDeleteRxSa (draft)	

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

Service ID [hex]	0x55	
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 EthSwt (EthSwt Config/EthSwtIdx), PortIdx in context of EthSwt (EthSwtPort/EthSwtPortIdx).
	An	Association Number to use in the MACsec's reception secure association
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: The request has been accepted `E_NOT_OK`: The request has not been accepted
Description	Request the Ethernet Switch Driver to remove the Reception Secure Association identified by the provided Association Number. **Tags:** atp.Status=draft	
Available via	EthSwt.h	

⌋

8.3.11.10 EthSwt_MacSecGetTxSaNextPn

[SWS_EthSwt_91133] Definition of API function EthSwt_MacSecGetTxSaNextPn

Status: DRAFT

⌈

Service Name	EthSwt_MacSecGetTxSaNextPn (draft)	
Syntax	`Std_ReturnType EthSwt_MacSecGetTxSaNextPn ( const EthSwt_MgmtInfoType* MgmtInfoPtr, uint8 An, uint64* NextPnPtr )`	
Service ID [hex]	0x56	
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 EthSwt (EthSwt Config/EthSwtIdx), PortIdx in context of EthSwt (EthSwtPort/EthSwtPortIdx).
	An	Association Number to use in the MACsec's reception secure association
Parameters (inout)	None	
Parameters (out)	NextPnPtr	Pointer to the Next Packet Number read out from the MACsec Entity (SecY)
Return value	Std_ReturnType	`E_OK`: The request has been accepted `E_NOT_OK`: The request has not been accepted
Description	Request the Ethernet Switch Driver to return the Packet Number that is used for the next packet in the given Transmission Secure Association. **Tags:** atp.Status=draft	
Available via	EthSwt.h	

⌋

8.3.11.11 EthSwt_MacSecGetMacSecStatistics

[SWS_EthSwt_91134] Definition of API function EthSwt_MacSecGetMacSecStatistics

Status: DRAFT

⌈

Service Name	EthSwt_MacSecGetMacSecStatistics (draft)
Syntax	`Std_ReturnType EthSwt_MacSecGetMacSecStatistics ( const EthSwt_MgmtInfoType* MgmtInfoPtr, Mka_Stats_SecYType* MacSecStatsPtr )`
Service ID [hex]	0x57
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 EthSwt (EthSwt Config/EthSwtIdx), PortIdx in context of EthSwt (EthSwtPort/Eth SwtPortIdx).
Parameters (inout)	None	
Parameters (out)	MacSecStatsPtr	Pointer to a structure including the MACsec statistics of an MKA participant
Return value	Std_ReturnType	`E_OK`: The request has been accepted `E_NOT_OK`: The request has not been accepted
Description	Request the Ethernet Switch Driver to provide MACsec statistics. **Tags:** atp.Status=draft	
Available via	EthSwt.h	

⌋

8.3.11.12 EthSwt_MacSecSetControlledPortEnabled

[SWS_EthSwt_91139] Definition of API function EthSwt_MacSecSetControlled PortEnabled

Status: DRAFT

⌈

Service Name	EthSwt_MacSecSetControlledPortEnabled (draft)	

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

Service ID [hex]	0x5c	
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 EthSwt (EthSwt Config/EthSwtIdx), PortIdx in context of EthSwt (EthSwtPort/Eth SwtPortIdx).
	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	EthSwt.h	

⌋

8.3.12 Streams

8.3.12.1 EthSwt_ExtractStreamHandleIdx

[SWS_EthSwt_91043] Definition of API function EthSwt_ExtractStreamHandleIdx

Status: DRAFT

⌈

Service Name	EthSwt_ExtractStreamHandleIdx (draft)	
Syntax	`Std_ReturnType EthSwt_ExtractStreamHandleIdx ( const uint8* DataPtr, uint16 LenByte, uint16* StreamHandleIdxPtr )`	
Service ID [hex]	0x5d	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	DataPtr	Pointer to payload of received Ethernet frame.
	LenByte	Length (bytes) of the payload in received frame.
Parameters (inout)	None	
Parameters (out)	StreamHandleIdxPtr	Pointer to the StreamHandleIdx extracted from the network packet
Return value	Std_ReturnType	E_OK: The operation was carried out successfully E_NOT_OK: The operation failed to execute
Description	Extracts the StreamHandleIdx from the switch vendor specific part of the network packet header **Tags:** atp.Status=draft	
Available via	EthSwt.h	

⌋

8.3.12.2 EthSwt_GetStreamStatistics

[SWS_EthSwt_91042] Definition of API function EthSwt_GetStreamStatistics

Status: DRAFT

⌈

Service Name	EthSwt_GetStreamStatistics (draft)	
Syntax	`void EthSwt_GetStreamStatistics ( uint8 SwitchIdx )`	
Service ID [hex]	0x5e	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	SwitchIdx	Index of the switch within the context of the Ethernet Switch Driver
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	

▽

△

Description	Requests the statistics (bucket counter values) of an Ethernet switch of all configured streams. **Tags:** atp.Status=draft
Available via	EthSwt.h

⌋

8.3.12.3 EthSwt_SetStreamState

[SWS_EthSwt_91041] Definition of API function EthSwt_SetStreamState

Status: DRAFT

⌈

Service Name	EthSwt_SetStreamState (draft)	

Syntax	`void EthSwt_SetStreamState (`
	`  uint8 SwitchIdx,`
	`  uint8 StreamHandleIdx,`
	`  boolean StreamActivityStatus`
	`)`	

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
	StreamHandleIdx	Pointer to the StreamHandleIdx for which the status shall be set
	StreamActivityStatus	Activity status of the StreamHandleIdx (True = active, False = inactive) to be set
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	This function is called by an upper layer application (e.g. diagnostic application) via the EthIf module to control the activity status of a configured stream (given with StreamHandleIdx) within an dedicated Ethernet switch (given with SwitchIdx). **Tags:** atp.Status=draft	
Available via	EthSwt.h	

⌋

8.4 Scheduled functions

8.4.1 EthSwt_MainFunction

[SWS_EthSwt_00114] Definition of scheduled function EthSwt_MainFunction

Upstream requirements: SRS_BSW_00433

⌈

Service Name	EthSwt_MainFunction

Syntax	`void EthSwt_MainFunction (`
	`  void`
	`)`

Service ID [hex]	0x1c
Description	Service to support asynchronous behavior of API calls
Available via	SchM_EthSwt.h

⌋

[SWS_EthSwt_00115]

Upstream requirements: SRS_BSW_00433, SRS_BSW_00478

⌈The `EthSwt_MainFunction` support asynchronous behavior of API calls. This function is directly called by Basic Software Scheduler.⌋

8.4.2 EthSwt_BackgroundTask

[SWS_EthSwt_91104] Definition of API function EthSwt_BackgroundTask ⌈

Service Name	EthSwt_BackgroundTask

Syntax	`void EthSwt_BackgroundTask (`
	`  void`
	`)`

Service ID [hex]	0x46
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Parameters (in)	None
Parameters (inout)	None
Parameters (out)	None
Return value	None
Description	The background task should be scheduled as often as possible when no other task runs. It may be used for switch and port initialization in case the EthSwt_Init function needs too much time.
Available via	SchM_EthSwt.h

⌋

8.5 Expected interfaces

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

8.5.1 Mandatory Interfaces

This chapter defines all external interfaces which are required to fulfill the core functionality of the module.

No mandatory Interfaces defined.

8.5.2 Optional Interfaces

This chapter defines all external interfaces which are required to fulfill an optional functionality of the module.

[SWS_EthSwt_00098] Definition of optional interfaces requested by module Eth Swt

Upstream requirements: SRS_Eth_00122, SRS_Eth_00118, SRS_Eth_00119, SRS_Eth_00120, SRS_Eth_00087, SRS_Eth_00125, SRS_BSW_00375

⌈

API Function	*Header File*	*Description*
Dem_SetEventStatus	Dem.h	Called by SW-Cs or BSW modules to report monitor status information to the Dem. BSW modules calling Dem_SetEventStatus can safely ignore the return value. This API will be available only if ({Dem/Dem ConfigSet/DemEventParameter/DemEvent ReportingType} == STANDARD_REPORTING)
Det_ReportError	Det.h	Service to report development errors.
EthIf_GetTransceiverMode	EthIf.h	Obtains the state of the indexed transceiver
EthIf_ReadMii	EthIf.h	Reads a transceiver register.
EthIf_ReadMmd	EthIf.h	Reads a transceiver register using Clause 45 access if supported by hardware or implements a Clause 45 access using Clause 22 operations.
EthIf_SetTransceiverMode	EthIf.h	Enables / disables the indexed transceiver
EthIf_StartAutoNegotiation	EthIf.h	Restarts the negotiation of the transmission parameters used by the indexed transceiver
EthIf_StreamStateIndication (draft)	EthIf_Cbk.h	The function is called by the EthSwt driver module once it has successfully set the streams activity status in the Ethernet switch given with SwitchIdx, triggered by a previous call of EthIf_SetStreamState. **Tags:** atp.Status=draft
EthIf_StreamStatisticsIndication (draft)	EthIf_Cbk.h	The function is called by the lower layer once it has successfully retrieved the stream statistics (i.e. bucket counter values) from the EthSwt driver given with SwitchIdx. **Tags:** atp.Status=draft
EthIf_TransceiverGetBaudRate	EthIf.h	Obtains the baud rate of the indexed transceiver
EthIf_TransceiverGetDuplexMode	EthIf.h	Obtains the duplex mode of the indexed transceiver
EthIf_TransceiverGetLinkState	EthIf.h	Obtains the link state of the indexed transceiver
EthIf_WriteMii	EthIf.h	Writes a transceiver register.

▽

△

API Function	Header File	Description
EthIf_WriteMmd	EthIf.h	Writes a transceiver register using Clause 45 access or implements a Clause 45 access using Clause 22 operations.
NvM_GetErrorStatus	NvM.h	Service to read the block dependent error/status information.
NvM_ReadBlock	NvM.h	Service to copy the data of the NV block to its corresponding RAM block.
NvM_WriteBlock	NvM.h	Service to copy the data of the RAM block to its corresponding NV block.
Spi_AsyncTransmit	Spi.h	Service to transmit data on the SPI bus.
Spi_Cancel	Spi.h	Service cancels the specified on-going sequence transmission.
Spi_ReadIB	Spi.h	Service for reading synchronously one or more data from an IB SPI Handler/Driver Channel specified by parameter.
Spi_SetAsyncMode	Spi.h	Service to set the asynchronous mechanism mode for SPI busses handled asynchronously.
Spi_SetupEB	Spi.h	Service to setup the buffers and the length of data for the EB SPI Handler/Driver Channel specified.
Spi_SyncTransmit	Spi.h	Service to transmit data on the SPI bus
Spi_WriteIB	Spi.h	Service for writing one or more data to an IB SPI Handler/Driver Channel specified by parameter.

⌋

[SWS_EthSwt_00192]

Upstream requirements: SRS_Eth_00122

⌈The NvM APIs will only be used if the respective block is not configured for `NvM_ReadAll` and `NvM_WriteAll`.⌋

8.5.3 Configurable interfaces

In this chapter all interfaces are listed where the target function could be configured.

The names of these kind of interfaces are not fixed because they are configurable.

8.5.3.1 <EthSwtLinkDownCallout>

[SWS_EthSwt_00117] Definition of callback function <EthSwtLinkDownCallout>

Upstream requirements: SRS_Eth_00119, SRS_Eth_00087

⌈

Service Name	<EthSwtLinkDownCallout>	
Syntax	`void <EthSwtLinkDownCallout> ( uint8 SwitchIdx, uint8 PortIdx )`	
Service ID [hex]	0x19	
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)	None	
Return value	None	
Description	Is called, if a link which is configured goes down.	
Available via	EthSwt_Externals.h	

⌋

[SWS_EthSwt_00118]

Upstream requirements: SRS_Eth_00119, SRS_Eth_00087

⌈The function `<EthSwtLinkDownCallout>` shall be called if a link, which is configured, goes down (link loss). The function provides the Switch index and the Port index, such that the port which went down can be identified.⌋

8.5.3.2 <EthSwtLinkUpCallout>

[SWS_EthSwt_00203] Definition of callback function <EthSwtLinkUpCallout>

Upstream requirements: SRS_Eth_00119, SRS_Eth_00087

⌈

Service Name	<EthSwtLinkUpCallout>	
Syntax	`void <EthSwtLinkUpCallout> ( uint8 SwitchIdx, uint8 PortIdx )`	
Service ID [hex]	0x1a	
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)	None
Return value	None
Description	Is called, if a link which is configured goes up
Available via	EthSwt_Externals.h

⌋

[SWS_EthSwt_00204]

Upstream requirements: SRS_Eth_00119, SRS_Eth_00087

⌈The function `<EthSwtLinkUpCallout>` shall be called if a link, which is configured, goes up. The function provides the Switch index and the Port index, such that the port which went up can be identified.⌋

Note: If the hardware cannot signal a link up with an interrupt, the status of the link has to be determined in polling mode by checking the state of the link.

8.5.3.3 <GetCfgDataRawDone>

[SWS_EthSwt_91032] Definition of callback function <GetCfgDataRawDone>

Upstream requirements: SRS_Eth_00123

⌈

Service Name	<GetCfgDataRawDone>	
Syntax	`void <GetCfgDataRawDone> ( uint8 SwitchIdx )`	
Service ID [hex]	0x43	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	SwitchIdx	Index of the Ethernet switch where the Configuration is read.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	The call of the function EthSwt_GetCfgDataRaw() triggers a asynchrony read of a certain memory section of the Ethernet switch driver. If the read is done, the configured callout function <GetCfgDataRawDone> shall be called]	
Available via	EthSwt_Externals.h	

⌋

8.6 Service Interfaces

No direct access is necessary from the application layer.

9 Sequence diagrams

9.1 Switch Management support

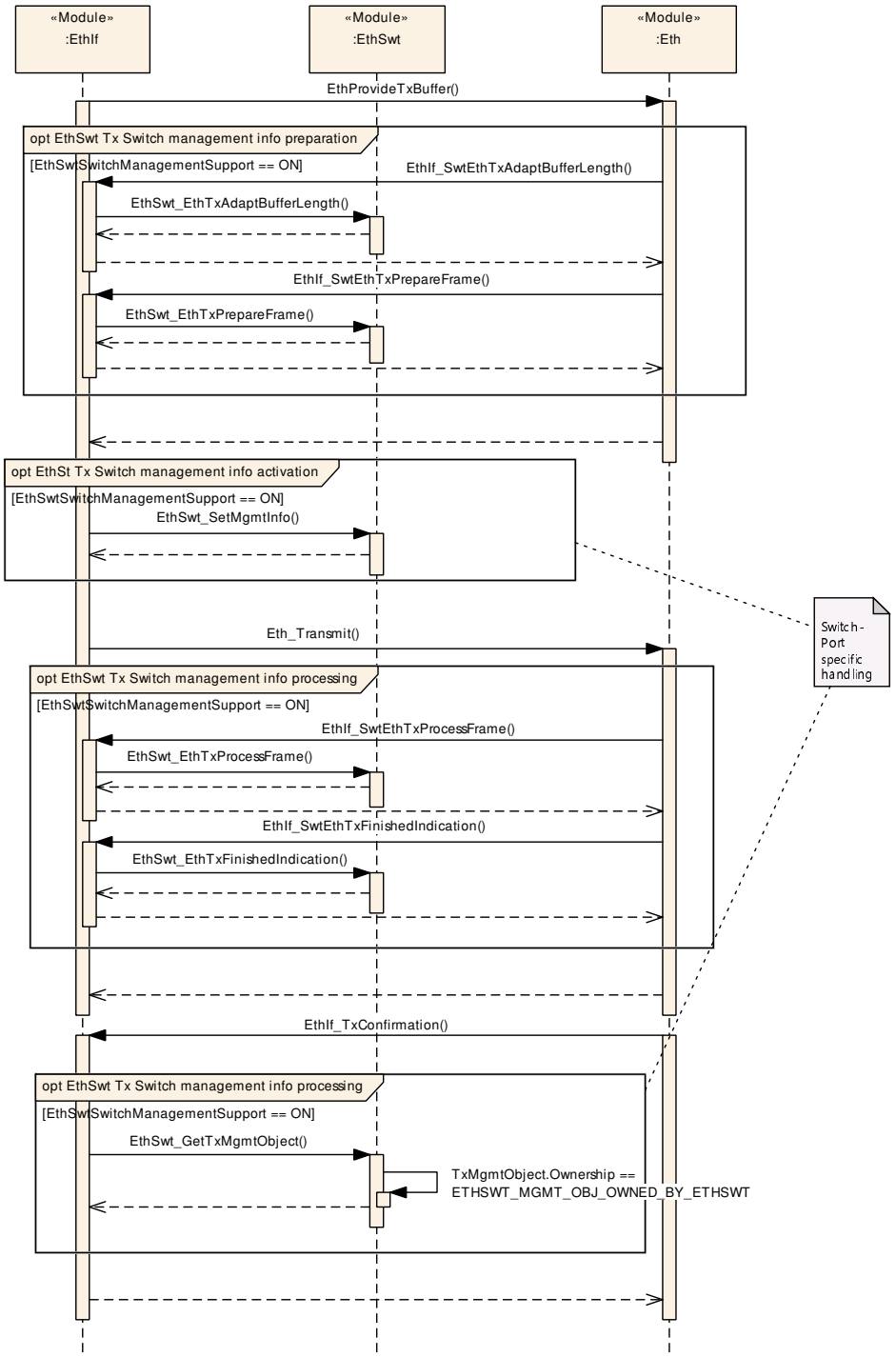

Figure 9.1: Switch Management support for transmission

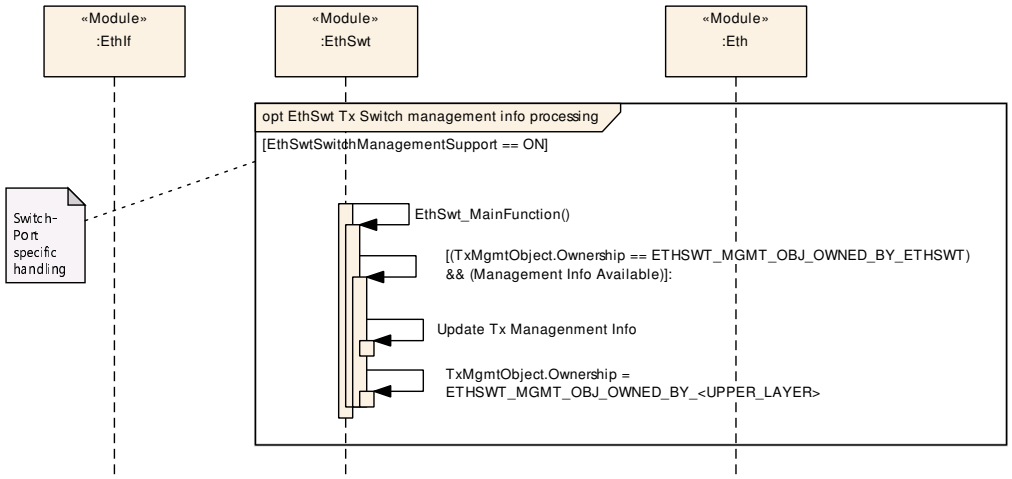

Figure 9.2: Switch Management support for transmission cont.

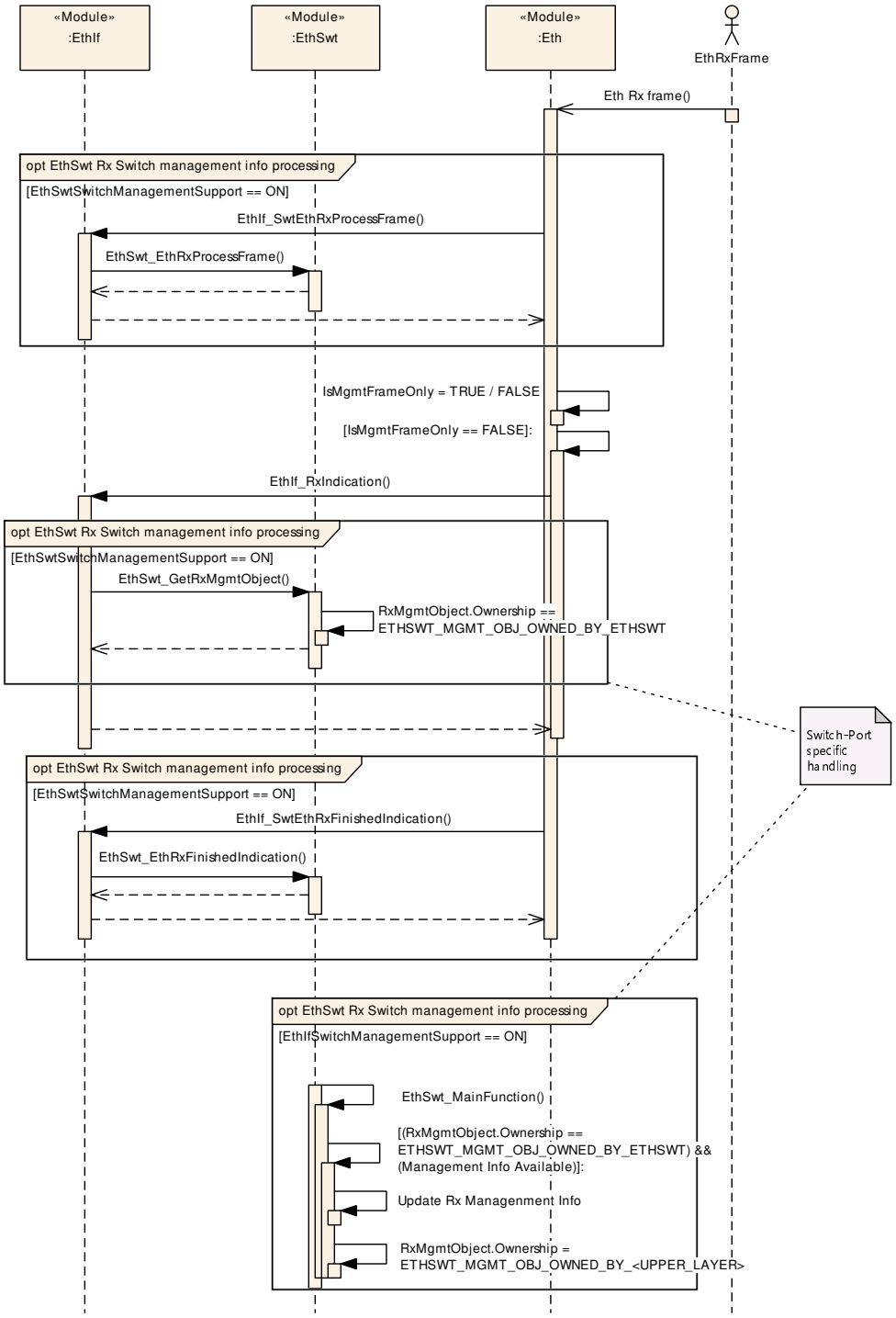

Figure 9.3: Management support for reception

10 Configuration specification

Chapter 10.2 specifies the structure (containers) and the parameters of the module EthSwt.

10.1 Containers and configuration parameters

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

[SWS_EthSwt_00414] ⌈The Ethernet Switch Driver module shall reject configurations with partition mappings which are not supported by the implementation.⌋

10.1.1 EthSwt

[ECUC_EthSwt_00046] Definition of EcucModuleDef EthSwt ⌈

Module Name	EthSwt
Description	Configuration of the EthSwt (Ethernet Switch Driver) module.
Post-Build Selectable Support	true
Post-Build Loadable Support	true

Included Containers		
Container Name	**Multiplicity**	**Description**
EthSwtConfig	1..*	Configuration of one Ethernet Switch.
EthSwtGeneral	1	General configuration of Ethernet Switch Driver module.

⌋

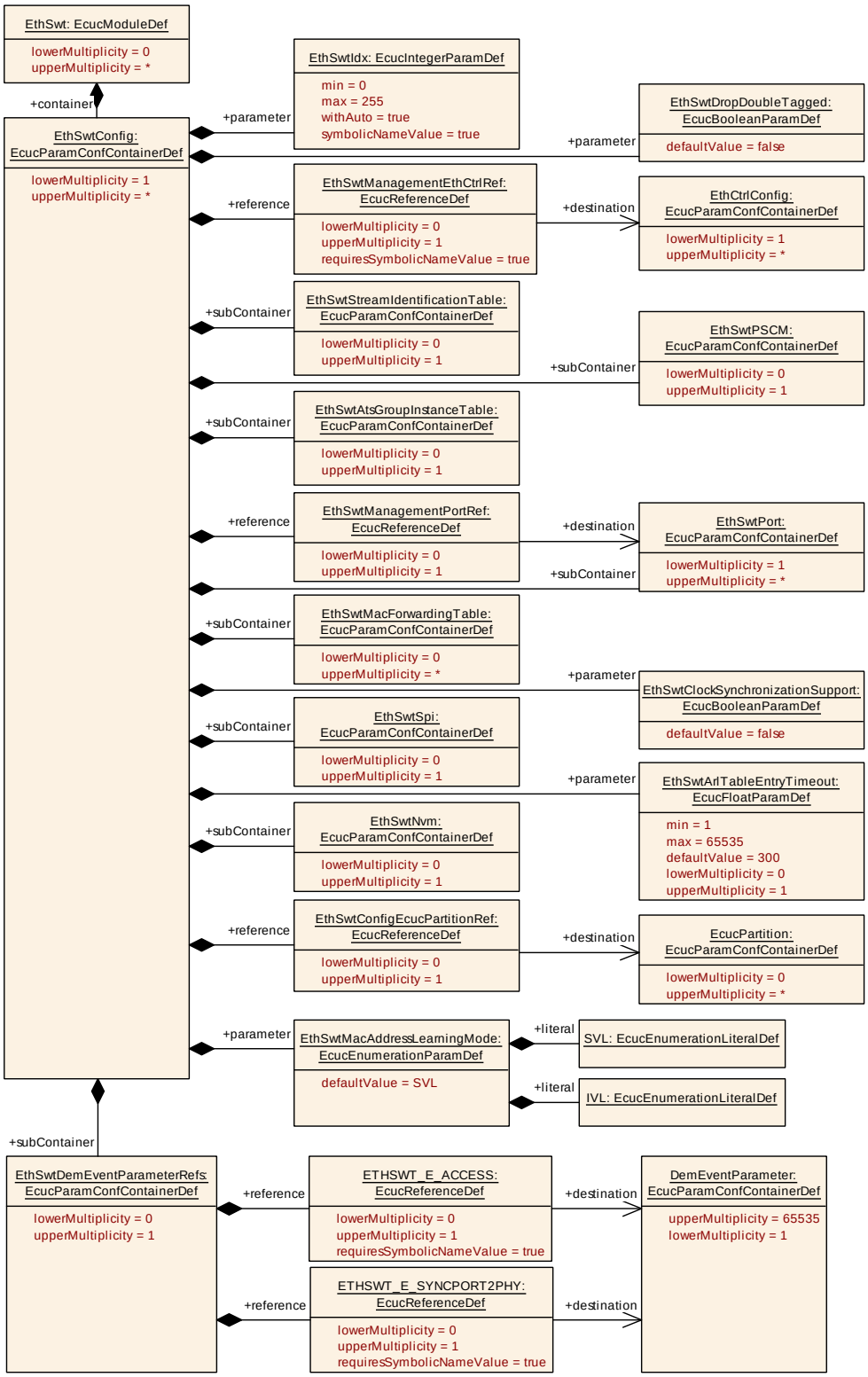

Figure 10.1: EthSwt

10.1.2 EthSwtGeneral

[ECUC_EthSwt_00003] Definition of EcucParamConfContainerDef EthSwtGeneral ⌈

Container Name	EthSwtGeneral
Parent Container	EthSwt
Upstream mapping defined	–
Description	General configuration of Ethernet Switch Driver module.
Multiplicity	1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
EthSwtCheckWakeupApi	1	[ECUC_EthSwt_00136]
EthSwtDeletePortMirrorConfigurationApi	1	[ECUC_EthSwt_00133]
EthSwtDevErrorDetect	1	[ECUC_EthSwt_00002]
EthSwtEnableCableDiagnosticApi	1	[ECUC_EthSwt_00135]
EthSwtEnableVlanApi	1	[ECUC_EthSwt_00055]
EthSwtGetArlTableApi	1	[ECUC_EthSwt_00052]
EthSwtGetBaudRateApi	1	[ECUC_EthSwt_00121]
EthSwtGetCfgDataRawDone	0..1	[ECUC_EthSwt_00124]
EthSwtGetCfgRaw	1	[ECUC_EthSwt_00123]
EthSwtGetCounterValuesApi	1	[ECUC_EthSwt_00053]
EthSwtGetDuplexModeApi	1	[ECUC_EthSwt_00122]
EthSwtGetLinkStateApi	1	[ECUC_EthSwt_00120]
EthSwtGetMacLearningModeApi	1	[ECUC_EthSwt_00061]
EthSwtGetMaxQueueBufferFillLevelApi	1	[ECUC_EthSwt_00131]
EthSwtGetPortCableDiagnosticsResultApi	1	[ECUC_EthSwt_00092]
EthSwtGetPortIdentifierApi	1	[ECUC_EthSwt_00083]
EthSwtGetPortMacAddrApi	1	[ECUC_EthSwt_00051]
EthSwtGetPortMacAddrVlanApi	0..1	[ECUC_EthSwt_00235]
EthSwtGetPortMirrorStateApi	1	[ECUC_EthSwt_00087]
EthSwtGetPortSignalQualityApi	1	[ECUC_EthSwt_00082]
EthSwtGetRxStatsApi	1	[ECUC_EthSwt_00065]
EthSwtGetSwitchIdentifierApi	1	[ECUC_EthSwt_00084]
EthSwtGetSwitchPortModeApi	1	[ECUC_EthSwt_00118]
EthSwtGetSwitchPortWakeupReasonApi	1	[ECUC_EthSwt_00137]
EthSwtGetSwitchRegApi	1	[ECUC_EthSwt_00066]
EthSwtGetTxErrorCounterValuesApi	1	[ECUC_EthSwt_00100]
EthSwtGetTxStatsApi	1	[ECUC_EthSwt_00099]
EthSwtGlobalTimeSupportApi	1	[ECUC_EthSwt_00107]
EthSwtIndex	1	[ECUC_EthSwt_00033]
EthSwtLinkDownCallout	0..1	[ECUC_EthSwt_00115]
EthSwtLinkUpCallout	0..1	[ECUC_EthSwt_00116]

▽

△

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
EthSwtLowPowerModeSupport	0..1	[ECUC_EthSwt_00102]
EthSwtMacSecApi	1	[ECUC_EthSwt_00260]
EthSwtMainFunctionMacSecPeriod	1	[ECUC_EthSwt_00261]
EthSwtMainFunctionPeriod	1	[ECUC_EthSwt_00071]
EthSwtManagementSupportApi	1	[ECUC_EthSwt_00108]
EthSwtPersistentConfigurationResult	1	[ECUC_EthSwt_00062]
EthSwtPersistentConfigurationResultCallback	0..1	[ECUC_EthSwt_00063]
EthSwtPublicCddHeaderFile	0..*	[ECUC_EthSwt_00064]
EthSwtReadMmdApi	1	[ECUC_EthSwt_00243]
EthSwtReadPortMirrorConfigurationApi	1	[ECUC_EthSwt_00086]
EthSwtReadTrcvRegisterApi	1	[ECUC_EthSwt_00069]
EthSwtResetConfigurationApi	1	[ECUC_EthSwt_00049]
EthSwtSetForwardingModeApi	1	[ECUC_EthSwt_00104]
EthSwtSetMacLearningModeApi	1	[ECUC_EthSwt_00060]
EthSwtSetPortLoopbackModeApi	1	[ECUC_EthSwt_00090]
EthSwtSetPortMirrorStateApi	1	[ECUC_EthSwt_00088]
EthSwtSetPortTestModeApi	1	[ECUC_EthSwt_00089]
EthSwtSetPortTxModeApi	1	[ECUC_EthSwt_00091]
EthSwtSetSwitchPortModeApi	1	[ECUC_EthSwt_00117]
EthSwtSetSwitchRegApi	1	[ECUC_EthSwt_00067]
EthSwtStartSwitchPortAutoNegotiationApi	1	[ECUC_EthSwt_00119]
EthSwtStoreConfigurationApi	1	[ECUC_EthSwt_00050]
EthSwtVerifyConfigApi	1	[ECUC_EthSwt_00105]
EthSwtWriteMmdApi	1	[ECUC_EthSwt_00244]
EthSwtWritePortMirrorConfigurationApi	1	[ECUC_EthSwt_00085]
EthSwtWriteTrcvRegisterApi	1	[ECUC_EthSwt_00070]
EthSwtEcucPartitionRef	0..*	[ECUC_EthSwt_00129]

No Included Containers

⌋

[ECUC_EthSwt_00136] Definition of EcucBooleanParamDef EthSwtCheckWakeupApi ⌈

Parameter Name	EthSwtCheckWakeupApi
Parent Container	EthSwtGeneral
Upstream mapping defined	–
Description	Enables / Disables EthSwt_CheckWakeup API.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_EthSwt_00133] Definition of EcucBooleanParamDef EthSwtDeletePortMirrorConfigurationApi ⌈

Parameter Name	EthSwtDeletePortMirrorConfigurationApi
Parent Container	EthSwtGeneral
Upstream mapping defined	–
Description	Enables / Disables EthSwt_DeletePortMirrorConfiguration API
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_EthSwt_00002] Definition of EcucBooleanParamDef EthSwtDevErrorDetect ⌈

Parameter Name	EthSwtDevErrorDetect
Parent Container	EthSwtGeneral
Upstream mapping defined	–
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
Pre-Compile only	true

⌋

[ECUC_EthSwt_00135] Definition of EcucBooleanParamDef EthSwtEnableCableDiagnosticApi ⌈

Parameter Name	EthSwtEnableCableDiagnosticApi
Parent Container	EthSwtGeneral
Upstream mapping defined	–
Description	Enable/disable the APIs for cable diagnostic: EthSwt_RunPortCableDiagnostic, EthSwt_GetPortCableDiagnosticsResult
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_EthSwt_00055] Definition of EcucBooleanParamDef EthSwtEnableVlanApi ⌈

Parameter Name	EthSwtEnableVlanApi
Parent Container	EthSwtGeneral
Upstream mapping defined	–
Description	Enables / Disables EthSwt_EnableVLAN API.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_EthSwt_00052] Definition of EcucBooleanParamDef EthSwtGetArlTableApi ⌈

Parameter Name	EthSwtGetArlTableApi
Parent Container	EthSwtGeneral
Upstream mapping defined	–
Description	Enables / Disables EthSwt_GetArlTable API.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_EthSwt_00121] Definition of EcucBooleanParamDef EthSwtGetBaudRateApi ⌈

Parameter Name	EthSwtGetBaudRateApi
Parent Container	EthSwtGeneral
Upstream mapping defined	–
Description	Enables / Disables EthSwt_GetBaudRate API
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_EthSwt_00124] Definition of EcucFunctionNameDef EthSwtGetCfgDataRawDone ⌈

Parameter Name	EthSwtGetCfgDataRawDone
Parent Container	EthSwtGeneral
Upstream mapping defined	–

▽

△

Description	Defines the function name for <GetCfgDataRawDone> Dependency: The function GetCfgDataRawDone shall only be configured if parameter EthSwtGetCfgRaw is set to TRUE.
Multiplicity	0..1
Type	EcucFunctionNameDef
Default value	–
Regular Expression	–
Pre-Compile only	true

⌋

[ECUC_EthSwt_00123] Definition of EcucBooleanParamDef EthSwtGetCfgRaw ⌈

Parameter Name	EthSwtGetCfgRaw
Parent Container	EthSwtGeneral
Upstream mapping defined	–
Description	Disable /Enable support of reading raw data from switch memory
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_EthSwt_00053] Definition of EcucBooleanParamDef EthSwtGetCounterValuesApi ⌈

Parameter Name	EthSwtGetCounterValuesApi
Parent Container	EthSwtGeneral
Upstream mapping defined	–
Description	Enables / Disables EthSwt_GetCounterValues API
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_EthSwt_00122] Definition of EcucBooleanParamDef EthSwtGetDuplexModeApi ⌈

Parameter Name	EthSwtGetDuplexModeApi
Parent Container	EthSwtGeneral
Upstream mapping defined	–
Description	Enables / Disables EthSwt_GetDuplexMode API
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–

▽

△

Pre-Compile only	true

⌋

[ECUC_EthSwt_00120] Definition of EcucBooleanParamDef EthSwtGetLinkStateApi ⌈

Parameter Name	EthSwtGetLinkStateApi
Parent Container	EthSwtGeneral
Upstream mapping defined	–
Description	Enables / Disables EthSwt_GetLinkState API
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_EthSwt_00061] Definition of EcucBooleanParamDef EthSwtGetMacLearningModeApi ⌈

Parameter Name	EthSwtGetMacLearningModeApi
Parent Container	EthSwtGeneral
Upstream mapping defined	–
Description	Enables / Disables EthSwt_GetMacLearningMode API.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_EthSwt_00131] Definition of EcucBooleanParamDef EthSwtGetMaxQueueBufferFillLevelApi ⌈

Parameter Name	EthSwtGetMaxQueueBufferFillLevelApi
Parent Container	EthSwtGeneral
Upstream mapping defined	–
Description	Enables / Disables EthSwt_GetMaxQueueBufferFillLevel API.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_EthSwt_00092] Definition of EcucBooleanParamDef EthSwtGetPortCableDiagnosticsResultApi ⌈

Parameter Name	EthSwtGetPortCableDiagnosticsResultApi
Parent Container	EthSwtGeneral
Upstream mapping defined	–
Description	Enables / Disables EthSwt_GetPortCableDiagnosticsResult API
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_EthSwt_00083] Definition of EcucBooleanParamDef EthSwtGetPortIdentifierApi ⌈

Parameter Name	EthSwtGetPortIdentifierApi
Parent Container	EthSwtGeneral
Upstream mapping defined	–
Description	Enables / Disables EthSwt_GetPortIdentifier API
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_EthSwt_00051] Definition of EcucBooleanParamDef EthSwtGetPortMacAddrApi ⌈

Parameter Name	EthSwtGetPortMacAddrApi
Parent Container	EthSwtGeneral
Upstream mapping defined	–
Description	Enables / Disables EthSwt_GetPortMacAddr API.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_EthSwt_00235] Definition of EcucBooleanParamDef EthSwtGetPortMacAddrVlanApi ⌈

Parameter Name	EthSwtGetPortMacAddrVlanApi
Parent Container	EthSwtGeneral
Upstream mapping defined	–
Description	Enables / Disables EthSwt_GetPortMacAddrVlan API.
Multiplicity	0..1

▽

△

Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_EthSwt_00087] Definition of EcucBooleanParamDef EthSwtGetPortMirrorStateApi ⌈

Parameter Name	EthSwtGetPortMirrorStateApi
Parent Container	EthSwtGeneral
Upstream mapping defined	–
Description	Enables / Disables EthSwt_GetPortMirrorState API
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_EthSwt_00082] Definition of EcucBooleanParamDef EthSwtGetPortSignalQualityApi ⌈

Parameter Name	EthSwtGetPortSignalQualityApi
Parent Container	EthSwtGeneral
Upstream mapping defined	–
Description	Enables / Disables EthSwt_GetPortSignalQuality API
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_EthSwt_00065] Definition of EcucBooleanParamDef EthSwtGetRxStatsApi ⌈

Parameter Name	EthSwtGetRxStatsApi
Parent Container	EthSwtGeneral
Upstream mapping defined	–
Description	Enables / Disables EthSwt_GetRxStats API.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_EthSwt_00084] Definition of EcucBooleanParamDef EthSwtGetSwitchIdentifierApi ⌈

Parameter Name	EthSwtGetSwitchIdentifierApi
Parent Container	EthSwtGeneral
Upstream mapping defined	–
Description	Enables / Disables EthSwt_GetSwitchIdentifier API
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_EthSwt_00118] Definition of EcucBooleanParamDef EthSwtGetSwitchPortModeApi ⌈

Parameter Name	EthSwtGetSwitchPortModeApi
Parent Container	EthSwtGeneral
Upstream mapping defined	–
Description	Enables / Disables EthSwt_GetSwitchPortMode API
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_EthSwt_00137] Definition of EcucBooleanParamDef EthSwtGetSwitchPortWakeupReasonApi ⌈

Parameter Name	EthSwtGetSwitchPortWakeupReasonApi
Parent Container	EthSwtGeneral
Upstream mapping defined	–
Description	Enables / Disables EthSwt_GetSwitchPortWakeupReason API.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_EthSwt_00066] Definition of EcucBooleanParamDef EthSwtGetSwitchRegApi ⌈

Parameter Name	EthSwtGetSwitchRegApi
Parent Container	EthSwtGeneral
Upstream mapping defined	–
Description	Enables / Disables EthSwt_GetSwitchReg API.
Multiplicity	1

▽

△

Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_EthSwt_00100] Definition of EcucBooleanParamDef EthSwtGetTxErrorCounterValuesApi ⌈

Parameter Name	EthSwtGetTxErrorCounterValuesApi
Parent Container	EthSwtGeneral
Upstream mapping defined	–
Description	Enables/Disables Eth_GetTxErrorCounterValues API.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

⌋

[ECUC_EthSwt_00099] Definition of EcucBooleanParamDef EthSwtGetTxStatsApi ⌈

Parameter Name	EthSwtGetTxStatsApi
Parent Container	EthSwtGeneral
Upstream mapping defined	–
Description	Enables/Disables Eth_GetTxStats API.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

⌋

[ECUC_EthSwt_00107] Definition of EcucBooleanParamDef EthSwtGlobalTimeSupportApi ⌈

Parameter Name	EthSwtGlobalTimeSupportApi
Parent Container	EthSwtGeneral
Upstream mapping defined	–
Description	Enables/Disables the Global Time APIs used amongst others by Global Time Synchronization over Ethernet.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_EthSwt_00033] Definition of EcucIntegerParamDef EthSwtIndex ⌈

Parameter Name	EthSwtIndex	
Parent Container	EthSwtGeneral	
Upstream mapping defined	–	
Description	Specifies the InstanceId of this module instance. If only one instance is present it shall have the Id 0.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 255	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_EthSwt_00115] Definition of EcucFunctionNameDef EthSwtLinkDownCallout ⌈

Parameter Name	EthSwtLinkDownCallout
Parent Container	EthSwtGeneral
Upstream mapping defined	–
Description	Defines the function name for the <EthSwtLinkDownCallout> callout.
Multiplicity	0..1
Type	EcucFunctionNameDef
Default value	–
Regular Expression	–
Pre-Compile only	true

⌋

[ECUC_EthSwt_00116] Definition of EcucFunctionNameDef EthSwtLinkUpCallout ⌈

Parameter Name	EthSwtLinkUpCallout
Parent Container	EthSwtGeneral
Upstream mapping defined	–
Description	Defines the function name for the <EthSwtLinkUpCallout> callout.
Multiplicity	0..1
Type	EcucFunctionNameDef
Default value	–
Regular Expression	–
Pre-Compile only	true

⌋

[ECUC_EthSwt_00102] Definition of EcucBooleanParamDef EthSwtLowPowerModeSupport ⌈

Parameter Name	EthSwtLowPowerModeSupport
Parent Container	EthSwtGeneral
Upstream mapping defined	–
Description	Disable / Enable support of low power mode.
Multiplicity	0..1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

⌋

[ECUC_EthSwt_00260] Definition of EcucBooleanParamDef EthSwtMacSecApi ⌈

Parameter Name	EthSwtMacSecApi
Parent Container	EthSwtGeneral
Upstream mapping defined	–
Description	Enables / Disables the MACsec API
Multiplicity	1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

⌋

[ECUC_EthSwt_00261] Definition of EcucFloatParamDef EthSwtMainFunctionMacSecPeriod ⌈

Parameter Name	EthSwtMainFunctionMacSecPeriod
Parent Container	EthSwtGeneral
Upstream mapping defined	–
Description	The cycle time of the periodic EthSwt_MainFunctionMacSec of EthSwt. Defined in seconds.
Multiplicity	1
Type	EcucFloatParamDef
Range	]0 .. INF[
Default value	–
Pre-Compile only	true

⌋

[ECUC_EthSwt_00071] Definition of EcucFloatParamDef EthSwtMainFunctionPeriod ⌈

Parameter Name	EthSwtMainFunctionPeriod
Parent Container	EthSwtGeneral
Upstream mapping defined	–
Description	The cycle time of the periodic main function of EthSwt. Defined in seconds .

▽

△

Multiplicity	1
Type	EcucFloatParamDef
Range	]0 .. INF[
Default value	–
Pre-Compile only	true

⌋

[ECUC_EthSwt_00108] Definition of EcucBooleanParamDef EthSwtManagementSupportApi ⌈

Parameter Name	EthSwtManagementSupportApi
Parent Container	EthSwtGeneral
Upstream mapping defined	–
Description	Enables/Disables the Switch management APIs to support a Switch-port specific communication attribute access.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_EthSwt_00062] Definition of EcucBooleanParamDef EthSwtPersistentConfigurationResult ⌈

Parameter Name	EthSwtPersistentConfigurationResult
Parent Container	EthSwtGeneral
Upstream mapping defined	–
Description	Enables / Disables the callback API <User>_PersistentConfigurationResult.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_EthSwt_00063] Definition of EcucFunctionNameDef EthSwtPersistentConfigurationResultCallback ⌈

Parameter Name	EthSwtPersistentConfigurationResultCallback
Parent Container	EthSwtGeneral
Upstream mapping defined	–
Description	Defines the function name for <EthSwtPersistentConfigurationResultCallback>.
Multiplicity	0..1
Type	EcucFunctionNameDef
Default value	–
Regular Expression	–

▽

△

Pre-Compile only	true

⌋

[ECUC_EthSwt_00064] Definition of EcucStringParamDef EthSwtPublicCddHeaderFile ⌈

Parameter Name	EthSwtPublicCddHeaderFile
Parent Container	EthSwtGeneral
Upstream mapping defined	–
Description	Defines header files for callback functions which shall be included in case of CDDs.
Multiplicity	0..*
Type	EcucStringParamDef
Default value	–
Length	1-32
Regular Expression	–
Pre-Compile only	true

⌋

[ECUC_EthSwt_00243] Definition of EcucBooleanParamDef EthSwtReadMmdApi ⌈

Parameter Name	EthSwtReadMmdApi
Parent Container	EthSwtGeneral
Upstream mapping defined	–
Description	Enables/Disables EthSwt_ReadMmd
Multiplicity	1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

⌋

[ECUC_EthSwt_00086] Definition of EcucBooleanParamDef EthSwtReadPortMirrorConfigurationApi ⌈

Parameter Name	EthSwtReadPortMirrorConfigurationApi
Parent Container	EthSwtGeneral
Upstream mapping defined	–
Description	Enables / Disables EthSwt_ReadPortMirrorConfiguration API
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_EthSwt_00069] Definition of EcucBooleanParamDef EthSwtReadTrcvRegisterApi ⌈

Parameter Name	EthSwtReadTrcvRegisterApi
Parent Container	EthSwtGeneral
Upstream mapping defined	–
Description	Enables / Disables EthSwt_ReadTrcvRegister API.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_EthSwt_00049] Definition of EcucBooleanParamDef EthSwtResetConfigurationApi ⌈

Parameter Name	EthSwtResetConfigurationApi
Parent Container	EthSwtGeneral
Upstream mapping defined	–
Description	Enables / Disables EthSwt_ResetConfiguration API.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_EthSwt_00104] Definition of EcucBooleanParamDef EthSwtSetForwardingModeApi ⌈

Parameter Name	EthSwtSetForwardingModeApi
Parent Container	EthSwtGeneral
Upstream mapping defined	–
Description	Enables /disables EthSwt_SetForwardingMode API.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

⌋

[ECUC_EthSwt_00060] Definition of EcucBooleanParamDef EthSwtSetMacLearningModeApi ⌈

Parameter Name	EthSwtSetMacLearningModeApi
Parent Container	EthSwtGeneral
Upstream mapping defined	–
Description	Enables / Disables EthSwt_SetMacLearningMode API.
Multiplicity	1

▽

△

Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_EthSwt_00090] Definition of EcucBooleanParamDef EthSwtSetPortLoopbackModeApi ⌈

Parameter Name	EthSwtSetPortLoopbackModeApi
Parent Container	EthSwtGeneral
Upstream mapping defined	–
Description	Enables / Disables EthSwt_SetPortLoopbackModeApi API
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_EthSwt_00088] Definition of EcucBooleanParamDef EthSwtSetPortMirrorStateApi ⌈

Parameter Name	EthSwtSetPortMirrorStateApi
Parent Container	EthSwtGeneral
Upstream mapping defined	–
Description	Enables / Disables EthSwt_SetPortMirrorState API
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_EthSwt_00089] Definition of EcucBooleanParamDef EthSwtSetPortTestModeApi ⌈

Parameter Name	EthSwtSetPortTestModeApi
Parent Container	EthSwtGeneral
Upstream mapping defined	–
Description	Enables / Disables EthSwt_SetPortTestMode API
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_EthSwt_00091] Definition of EcucBooleanParamDef EthSwtSetPortTxModeApi ⌈

Parameter Name	EthSwtSetPortTxModeApi
Parent Container	EthSwtGeneral
Upstream mapping defined	–
Description	Enables / Disables EthSwt_SetPortTxModeApi API
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_EthSwt_00117] Definition of EcucBooleanParamDef EthSwtSetSwitchPortModeApi ⌈

Parameter Name	EthSwtSetSwitchPortModeApi
Parent Container	EthSwtGeneral
Upstream mapping defined	–
Description	Enables / Disables EthSwt_SetSwitchPortMode API
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_EthSwt_00067] Definition of EcucBooleanParamDef EthSwtSetSwitchRegApi ⌈

Parameter Name	EthSwtSetSwitchRegApi
Parent Container	EthSwtGeneral
Upstream mapping defined	–
Description	Enables / Disables EthSwt_SetSwitchReg API.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_EthSwt_00119] Definition of EcucBooleanParamDef EthSwtStartSwitchPortAutoNegotiationApi ⌈

Parameter Name	EthSwtStartSwitchPortAutoNegotiationApi
Parent Container	EthSwtGeneral
Upstream mapping defined	–
Description	Enables / Disables EthSwt_StartSwitchPortAutoNegotiation API
Multiplicity	1

▽

△

Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_EthSwt_00050] Definition of EcucBooleanParamDef EthSwtStoreConfigurationApi ⌈

Parameter Name	EthSwtStoreConfigurationApi
Parent Container	EthSwtGeneral
Upstream mapping defined	–
Description	Enables / Disables EthSwt_StoreConfiguration API.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_EthSwt_00105] Definition of EcucBooleanParamDef EthSwtVerifyConfigApi ⌈

Parameter Name	EthSwtVerifyConfigApi
Parent Container	EthSwtGeneral
Upstream mapping defined	–
Description	Enables /disables EthSwt_VerifyConfig API.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

⌋

[ECUC_EthSwt_00244] Definition of EcucBooleanParamDef EthSwtWriteMmdApi ⌈

Parameter Name	EthSwtWriteMmdApi
Parent Container	EthSwtGeneral
Upstream mapping defined	–
Description	Enables/Disables EthSwt_WriteMmd
Multiplicity	1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

⌋

[ECUC_EthSwt_00085] Definition of EcucBooleanParamDef EthSwtWritePortMirrorConfigurationApi ⌈

Parameter Name	EthSwtWritePortMirrorConfigurationApi
Parent Container	EthSwtGeneral
Upstream mapping defined	–
Description	Enables / Disables EthSwt_WritePortMirrorConfiguration API
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_EthSwt_00070] Definition of EcucBooleanParamDef EthSwtWriteTrcvRegisterApi ⌈

Parameter Name	EthSwtWriteTrcvRegisterApi
Parent Container	EthSwtGeneral
Upstream mapping defined	–
Description	Enables / Disables EthSwt_WriteTrcvRegister API.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_EthSwt_00129] Definition of EcucReferenceDef EthSwtEcucPartitionRef ⌈

Parameter Name	EthSwtEcucPartitionRef	
Parent Container	EthSwtGeneral	
Upstream mapping defined	–	
Description	Maps the Ethernet switch driver to zero or multiple ECUC partitions to make the modules API available in this partition.	
Multiplicity	0..*	
Type	Reference to EcucPartition	
Post-Build Multiplicity	-	Selectable
Post-Build Value	-	Selectable

⌋

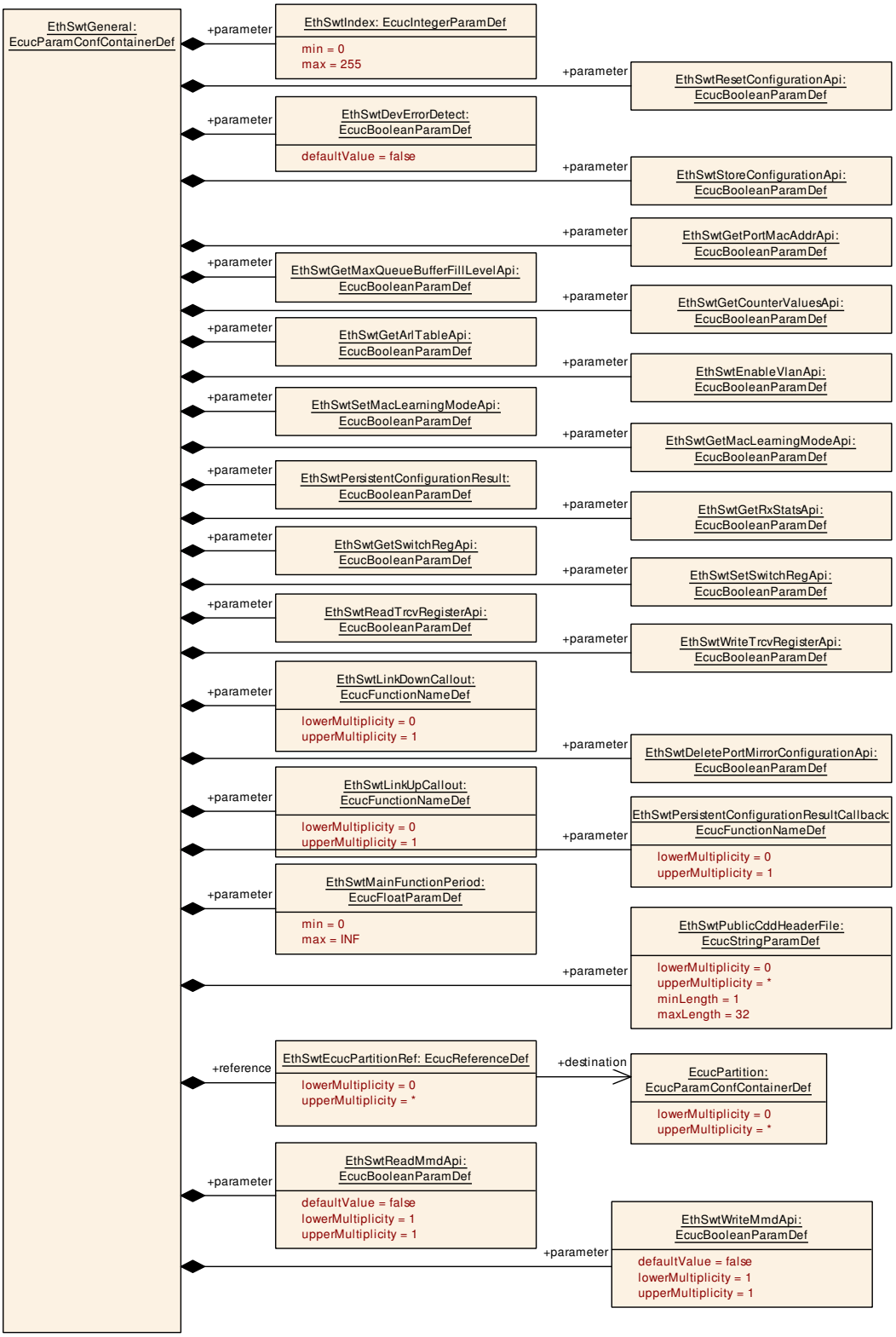

Figure 10.2: `EthSwtGeneral` (1/2)

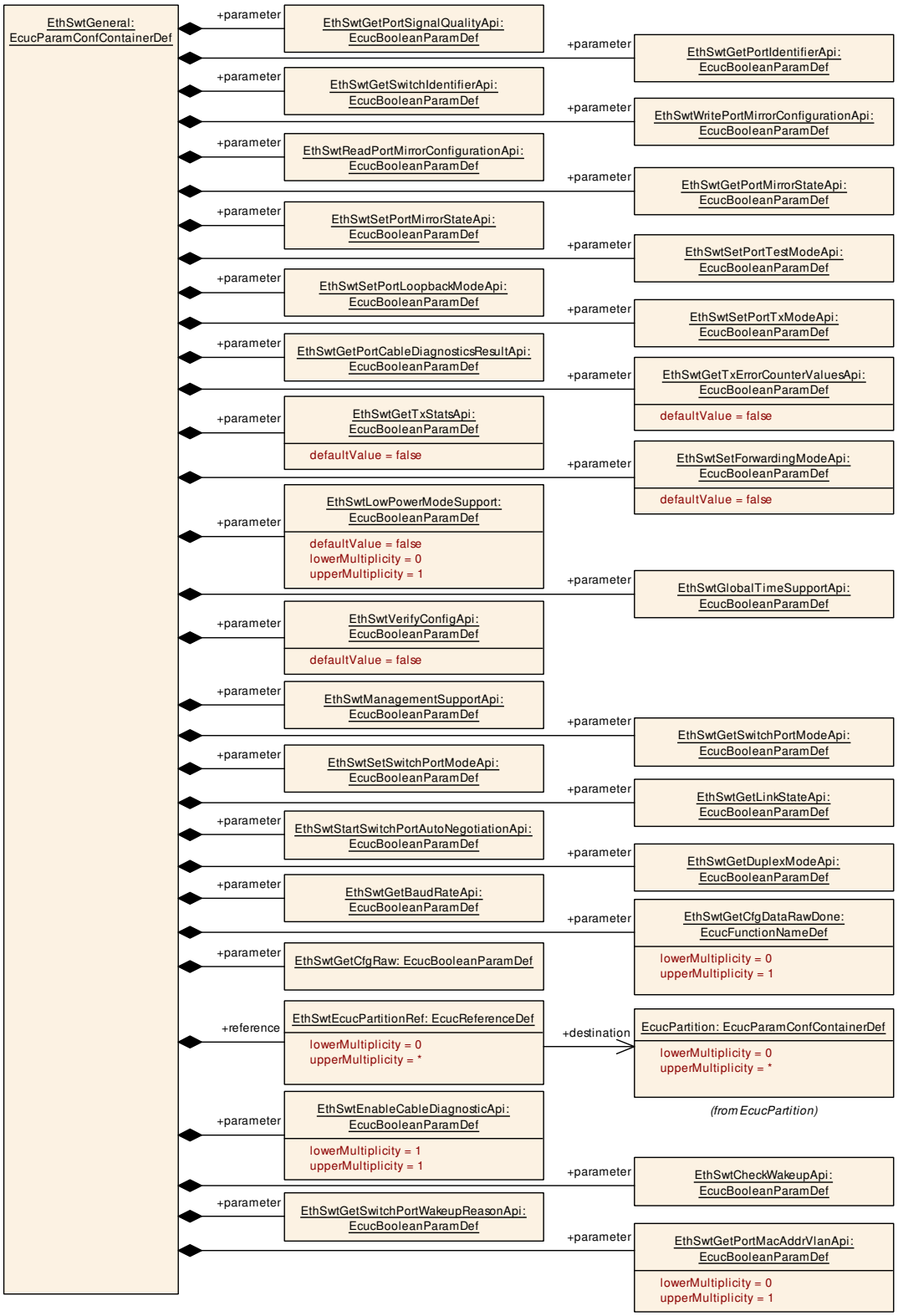

Figure 10.3: `EthSwtGeneral` (2/2)

10.1.3 EthSwtConfig

[ECUC_EthSwt_00001] Definition of EcucParamConfContainerDef EthSwtConfig
⌈

Container Name	EthSwtConfig	
Parent Container	EthSwt	
Upstream mapping defined	[14, CP_TPS_SystemTemplate]	
Description	Configuration of one Ethernet Switch.	
Multiplicity	1..*	
Post-Build Multiplicity	Loadable	Selectable
Configuration Parameters		

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
EthSwtArlTableEntryTimeout	0..1	[ECUC_EthSwt_00127]
EthSwtClockSynchronizationSupport	1	[ECUC_EthSwt_00128]
EthSwtDropDoubleTagged	1	[ECUC_EthSwt_00073]
EthSwtIdx	1	[ECUC_EthSwt_00004]
EthSwtMacAddressLearningMode	1	[ECUC_EthSwt_00236]
EthSwtUsedInternalPriorityUpperValue	1	[ECUC_EthSwt_00245]
EthSwtUsedTrafficClassUpperValue	1	[ECUC_EthSwt_00246]
EthSwtConfigEcucPartitionRef	0..1	[ECUC_EthSwt_00130]
EthSwtManagementEthCtrlRef	0..1	[ECUC_EthSwt_00110]
EthSwtManagementPortRef	0..1	[ECUC_EthSwt_00111]

Included Containers		
Container Name	**Multiplicity**	**Description**
EthSwtAtsGroupInstanceTable	0..1	Collection of AtsGroupInstanceEntrys. **Tags:** atp.Status=draft
EthSwtDemEventParameterRefs	0..1	Container for the references to DemEventParameter elements which shall be invoked using the API Dem_SetEventStatus in case the corresponding error occurs. The EventId is taken from the referenced DemEventParameter's DemEventId symbolic value. The standardized errors are provided in this container and can be extended by vendor-specific error references.
EthSwtMacForwardingTable	0..*	Represents a MAC forwarding table.
EthSwtNvm	0..1	Configuration of one Ethernet Switch Nvm usage in case the module requires non volatile memory in the Ecu to store switch configuration.
EthSwtPSCM	0..1	Per stream classification and metering. **Tags:** atp.Status=draft
EthSwtPort	1..*	Configuration of one Ethernet Switch Port.
EthSwtSpi	0..1	Configuration of one Ethernet Switch SPI access (if SPI is used).
EthSwtStreamIdentificationTable	0..1	Configuration of a stream identification table. **Tags:** atp.Status=draft
EthSwtUnknownMacDestAddressConfig	1..*	Definition to which EthSwtPorts an Ethernet frame shall be forwarded if the destination MAC address is not present in the address resolution lookup (ARL) table.

▽

△

Included Containers		
Container Name	**Multiplicity**	**Description**
EthSwtVlanMembership	0..4095	Determines the membership of this Ethernet switch and the referenced ports to the virtual network, i.e. frames with this VID can be received and transmitted via the referenced ports.

⌋

[ECUC_EthSwt_00127] Definition of EcucFloatParamDef EthSwtArlTableEntryTimeout ⌈

Parameter Name	EthSwtArlTableEntryTimeout	
Parent Container	EthSwtConfig	
Upstream mapping defined	–	
Description	If present, this parameter specifies the timeout in seconds for removing unused entries from the ARL table of the Ethernet switch. If the parameter is not configured, entries are not removed automatically.	
Multiplicity	0..1	
Type	EcucFloatParamDef	
Range	[1 .. 65535]	
Default value	300	
Pre-Compile only	true	

⌋

[ECUC_EthSwt_00128] Definition of EcucBooleanParamDef EthSwtClockSynchronizationSupport ⌈

Parameter Name	EthSwtClockSynchronizationSupport	
Parent Container	EthSwtConfig	
Upstream mapping defined	–	
Description	This parameter defines, if a Ethernet switch shall enable clock synchronization with another Ethernet switch to which it is connected via uplink port. If this parameter is set to TRUE the clock synchronization between connected Ethernet switches is activated and the clocks of the Ethernet switches are synchronized. If this parameter is set to FALSE the clock synchronization between connected Ethernet switches is deactivated. This parameter shall only be set to TRUE if the Ethernet switch hardware supports clock synchronization.	
Multiplicity	1	
Type	EcucBooleanParamDef	
Default value	false	
Post-Build Value	Loadable	Selectable

⌋

[ECUC_EthSwt_00073] Definition of EcucBooleanParamDef EthSwtDropDoubleTagged ⌈

Parameter Name	EthSwtDropDoubleTagged	
Parent Container	EthSwtConfig	
Upstream mapping defined	–	
Description	This parameter defines if a switch shall drop double tagged (Q in Q) frames. If this parameter is set to TRUE double tagged frames are dropped at all ports. If this parameter is set to FALSE, then double tagged frames are forwarded. If double tagging is used as a feature, this parameter must be set to FALSE. This parameter shall only be set to TRUE when Switch-HW supports the filtering of double tagged frames as filtering by SW is NOT possible!	
Multiplicity	1	
Type	EcucBooleanParamDef	
Default value	false	
Post-Build Value	Loadable	Selectable

⌋

[ECUC_EthSwt_00004] Definition of EcucIntegerParamDef EthSwtIdx ⌈

Parameter Name	EthSwtIdx	
Parent Container	EthSwtConfig	
Upstream mapping defined	–	
Description	Specifies the instance ID of the configured Ethernet Switch.	
Multiplicity	1	
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)	
Range	0 .. 255	
Default value	–	
With Auto	true	
Pre-Compile only	true	

⌋

[ECUC_EthSwt_00236] Definition of EcucEnumerationParamDef EthSwtMacAddressLearningMode ⌈

Parameter Name	EthSwtMacAddressLearningMode	
Parent Container	EthSwtConfig	
Upstream mapping defined	[14, CP_TPS_SystemTemplate]	
Description	Defines the MAC address learning mode specified by [7, IEEE802.1Q] either shared VLAN learning (SVL) or independent VLAN learning (IVL).	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	IVL	The Ethernet switch updates the ARL table (address resolution table) with an entry consiting of source MAC address and VLAN-ID of the received Ethernet frame and the ingress port from on which the Ethernet frame was received.

▽

△

	SVL	The Ethernet switch updates the ARL table (address resolution table) with an entry consiting of source MAC address of the received Ethernet frame and the ingress port on which the Ethernet frame was received.
Default value	SVL	
Pre-Compile only	true	

⌋

[ECUC_EthSwt_00245] Definition of EcucIntegerParamDef EthSwtUsedInternalPriorityUpperValue

Status: DRAFT

⌈

Parameter Name	EthSwtUsedInternalPriorityUpperValue	
Parent Container	EthSwtConfig	
Upstream mapping defined	[14, CP_TPS_SystemTemplate]	
Description	Configure the upper value of the used internal priority range in the Ethernet switch. The range of used internal priority values is defined from 0 to EthSwtUsedInternalPriorityUpperValue, where zero is the lowest and EthSwtUsedInternalPriorityUpperValue the highest internal priority value. Each priority is mapped to at least one traffic class. **Tags:** atp.Status=draft	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 4294967295	
Default value	7	
Pre-Compile only	true	

⌋

[ECUC_EthSwt_00246] Definition of EcucIntegerParamDef EthSwtUsedTrafficClassUpperValue

Status: DRAFT

⌈

Parameter Name	EthSwtUsedTrafficClassUpperValue	
Parent Container	EthSwtConfig	
Upstream mapping defined	[14, CP_TPS_SystemTemplate]	
Description	Configure the upper value of the used traffic class range in the Ethernet switch. The range of used traffic classes is defined from 0 to EthSwtUsedTrafficClassUpperValue. A traffic class is associated with exactly one egress queue at an egress port. **Tags:** atp.Status=draft	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 65535	

▽

△

Default value	7
Pre-Compile only	true

⌋

[ECUC_EthSwt_00130] Definition of EcucReferenceDef EthSwtConfigEcucPartitionRef ⌈

Parameter Name	EthSwtConfigEcucPartitionRef	
Parent Container	EthSwtConfig	
Upstream mapping defined	–	
Description	Maps the configuration of one single Ethernet switch to zero or one ECUC partitions. The ECUC partition referenced is a subset of the ECUC partitions where the Ethernet switch driver is mapped to.	
Multiplicity	0..1	
Type	Reference to EcucPartition	
Post-Build Multiplicity	-	Selectable
Post-Build Value	-	Selectable

⌋

[ECUC_EthSwt_00110] Definition of EcucReferenceDef EthSwtManagementEthCtrlRef ⌈

Parameter Name	EthSwtManagementEthCtrlRef
Parent Container	EthSwtConfig
Upstream mapping defined	–
Description	Reference to the Ethernet controller connected to the management port where the management frames will be transmitted/received.
Multiplicity	0..1
Type	Symbolic name reference to EthCtrlConfig
Pre-Compile only	true

⌋

[ECUC_EthSwt_00111] Definition of EcucReferenceDef EthSwtManagementPortRef ⌈

Parameter Name	EthSwtManagementPortRef
Parent Container	EthSwtConfig
Upstream mapping defined	–
Description	Reference to the port where the management CPU is connected to.
Multiplicity	0..1
Type	Reference to EthSwtPort
Pre-Compile only	true

⌋

10.1.4 EthSwtAtsGroupInstanceTable

[ECUC_EthSwt_00229] Definition of EcucParamConfContainerDef EthSwtAtsGroupInstanceTable

Status: DRAFT

⌈

Container Name	EthSwtAtsGroupInstanceTable	
Parent Container	EthSwtConfig	
Upstream mapping defined	–	
Description	Collection of AtsGroupInstanceEntrys. **Tags:** atp.Status=draft	
Multiplicity	0..1	
Post-Build Multiplicity	Loadable	Selectable
Configuration Parameters		

No Included Parameters

Included Containers		
Container Name	**Multiplicity**	**Description**
EthSwtAtsGroupInstanceEntry	0..*	AtsGroupInstanceEntry. **Tags:** atp.Status=draft

⌋

10.1.5 EthSwtAtsGroupInstanceEntry

[ECUC_EthSwt_00230] Definition of EcucParamConfContainerDef EthSwtAtsGroupInstanceEntry

Status: DRAFT

⌈

Container Name	EthSwtAtsGroupInstanceEntry	
Parent Container	EthSwtAtsGroupInstanceTable	
Upstream mapping defined	[14, CP_TPS_SystemTemplate]	
Description	AtsGroupInstanceEntry. **Tags:** atp.Status=draft	
Multiplicity	0..*	
Post-Build Multiplicity	Loadable	Selectable
Configuration Parameters		

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
EthSwtAtsGroupMaximumResidenceTime	1	[ECUC_EthSwt_00195]

No Included Containers

⌋

[ECUC_EthSwt_00195] Definition of EcucFloatParamDef EthSwtAtsGroupMaximumResidenceTime

Status: DRAFT

⌈

Parameter Name	EthSwtAtsGroupMaximumResidenceTime	
Parent Container	EthSwtAtsGroupInstanceEntry	
Upstream mapping defined	–	
Description	The parameter defines the maximum duration limit for which frames can reside in a bridge in seconds. **Tags:** atp.Status=draft	
Multiplicity	1	
Type	EcucFloatParamDef	
Range	]0 .. INF[	
Default value	–	
Post-Build Value	Loadable	Selectable

⌋

10.1.6 EthSwtDemEventParameterRefs

[ECUC_EthSwt_00016] Definition of EcucParamConfContainerDef EthSwtDemEventParameterRefs ⌈

Container Name	EthSwtDemEventParameterRefs	
Parent Container	EthSwtConfig	
Upstream mapping defined	–	
Description	Container for the references to DemEventParameter elements which shall be invoked using the API Dem_SetEventStatus in case the corresponding error occurs. The Event Id is taken from the referenced DemEventParameter's DemEventId symbolic value. The standardized errors are provided in this container and can be extended by vendor-specific error references.	
Multiplicity	0..1	
Post-Build Multiplicity	-	Selectable
Configuration Parameters		

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
ETHSWT_E_ACCESS	0..1	[ECUC_EthSwt_00006]
ETHSWT_E_SYNCPORT2PHY	0..1	[ECUC_EthSwt_00125]

No Included Containers

⌋

[ECUC_EthSwt_00006] Definition of EcucReferenceDef ETHSWT_E_ACCESS ⌈

Parameter Name	ETHSWT_E_ACCESS
Parent Container	EthSwtDemEventParameterRefs
Upstream mapping defined	–
Description	Reference to the DemEventParameter which shall be issued when the error "Ethernet Switch Access Failure" has occurred.
Multiplicity	0..1
Type	Symbolic name reference to DemEventParameter
Pre-Compile only	true

⌋

[ECUC_EthSwt_00125] Definition of EcucReferenceDef ETHSWT_E_SYNCPORT2PHY ⌈

Parameter Name	ETHSWT_E_SYNCPORT2PHY
Parent Container	EthSwtDemEventParameterRefs
Upstream mapping defined	–
Description	Reference to the DemEventParameter which shall be issued when the error "Ethernet switch port and the referenced Ethernet transceiver are in contradicting modes" has occurred.
Multiplicity	0..1
Type	Symbolic name reference to DemEventParameter
Pre-Compile only	true

⌋

10.1.7 EthSwtMacForwardingTable

[ECUC_EthSwt_00205] Definition of EcucParamConfContainerDef EthSwtMacForwardingTable ⌈

Container Name	EthSwtMacForwardingTable	
Parent Container	EthSwtConfig	
Upstream mapping defined	[14, CP_TPS_SystemTemplate]	
Description	Represents a MAC forwarding table.	
Multiplicity	0..*	
Post-Build Multiplicity	Loadable	Selectable
Configuration Parameters		

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
EthSwtPredefinedMacAddress	1	[ECUC_EthSwt_00206]
EthSwtMacForwardingTablePortRef	1..255	[ECUC_EthSwt_00207]
EthSwtVlanMembershipRef	0..4095	[ECUC_EthSwt_00237]

No Included Containers

⌋

[ECUC_EthSwt_00206] Definition of EcucStringParamDef EthSwtPredefinedMac Address ⌈

Parameter Name	EthSwtPredefinedMacAddress	
Parent Container	EthSwtMacForwardingTable	
Upstream mapping defined	[14, CP_TPS_SystemTemplate]	
Description	Specifies a 48-bit physical addresses (MAC addresses) network byte order, which can be reached via the referenced port and if available via the referenced VLAN . Note that further addresses can be learned during runtime.	
Multiplicity	1	
Type	EcucStringParamDef	
Default value	–	
Length	17-17	
Regular Expression	([0-9a-fA-F]{2}:){5}[0-9a-fA-F]{2}	
Post-Build Value	Loadable	Selectable

⌋

[ECUC_EthSwt_00207] Definition of EcucReferenceDef EthSwtMacForwarding TablePortRef ⌈

Parameter Name	EthSwtMacForwardingTablePortRef	
Parent Container	EthSwtMacForwardingTable	
Upstream mapping defined	[14, CP_TPS_SystemTemplate]	
Description	References the ports the MAC shall be assigned to.	
Multiplicity	1..255	
Type	Reference to EthSwtPort	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

⌋

[ECUC_EthSwt_00237] Definition of EcucReferenceDef EthSwtVlanMembership Ref ⌈

Parameter Name	EthSwtVlanMembershipRef	
Parent Container	EthSwtMacForwardingTable	
Upstream mapping defined	[14, CP_TPS_SystemTemplate]	
Description	References the VLAN-IDs the MAC address shall be assigned to. Please note, this reference is used if EthSwtMacAddressLearningMode is set to IVL (independent VLAN learning). Dependency: Those references are only valid if EthSwtMacAddressLearningMode is set to IVL and the EthSwtPort, which is referenced via EthSwtMacForwardingTablePort Ref, is member of the referenced VLAN.	
Multiplicity	0..4095	
Type	Reference to EthSwtVlanMembership	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

⌋

10.1.8 EthSwtNvm

[ECUC_EthSwt_00043] Definition of EcucParamConfContainerDef EthSwtNvm ⌈

Container Name	EthSwtNvm	
Parent Container	EthSwtConfig	
Upstream mapping defined	–	
Description	Configuration of one Ethernet Switch Nvm usage in case the module requires non volatile memory in the Ecu to store switch configuration.	
Multiplicity	0..1	
Post-Build Multiplicity	-	Selectable
Configuration Parameters		

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
EthSwtConfigurationNvmBlockDescriptorRef	1	[ECUC_EthSwt_00134]
EthSwtTableNvmBlockDescriptorRef	1	[ECUC_EthSwt_00044]

No Included Containers

⌋

[ECUC_EthSwt_00134] Definition of EcucReferenceDef EthSwtConfiguration NvmBlockDescriptorRef ⌈

Parameter Name	EthSwtConfigurationNvmBlockDescriptorRef
Parent Container	EthSwtNvm
Upstream mapping defined	–
Description	Reference to the Nvm block description in the Nvm module configuration to store e.g. the port mirror configurations
Multiplicity	1
Type	Symbolic name reference to NvMBlockDescriptor
Pre-Compile only	true

⌋

[ECUC_EthSwt_00044] Definition of EcucReferenceDef EthSwtTableNvmBlock DescriptorRef ⌈

Parameter Name	EthSwtTableNvmBlockDescriptorRef	
Parent Container	EthSwtNvm	
Upstream mapping defined	–	
Description	Reference to the Nvm block description in the Nvm module configuration to store e.g. the learned ARL table	
Multiplicity	1	
Type	Symbolic name reference to NvMBlockDescriptor	
Post-Build Value	-	Selectable

⌋

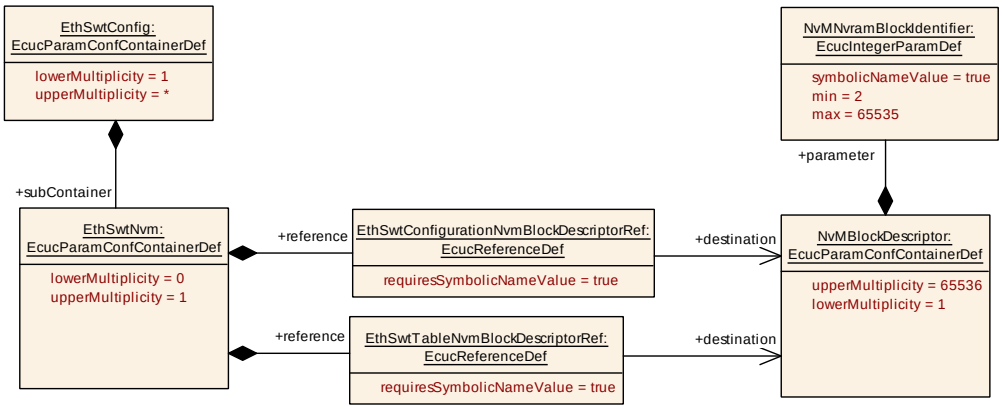

Figure 10.4: `EthSwtNvm`

10.1.9 EthSwtPSCM

[ECUC_EthSwt_00218] Definition of EcucParamConfContainerDef EthSwtPSCM

Status: DRAFT

⌈

Container Name	EthSwtPSCM	
Parent Container	EthSwtConfig	
Upstream mapping defined	–	
Description	Per stream classification and metering. **Tags:** atp.Status=draft	
Multiplicity	0..1	
Post-Build Multiplicity	Loadable	Selectable
Configuration Parameters		

No Included Parameters

Included Containers		
Container Name	**Multiplicity**	**Description**
EthSwtAtsInstanceTable	0..1	EthSwtAtsInstanceTable **Tags:** atp.Status=draft
EthSwtPSFP	0..1	Configuration of Per-stream Filtering and Policing (PSFP). **Tags:** atp.Status=draft

⌋

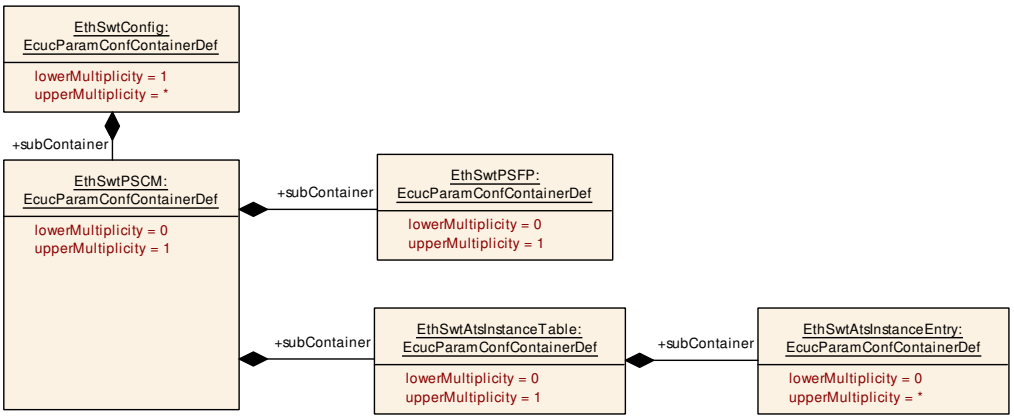

Figure 10.5: EthSwtPSCM

10.1.10 EthSwtAtsInstanceTable

[ECUC_EthSwt_00226] Definition of EcucParamConfContainerDef EthSwtAtsInstanceTable

Status: DRAFT

⌈

Container Name	EthSwtAtsInstanceTable
Parent Container	EthSwtPSCM
Upstream mapping defined	–
Description	EthSwtAtsInstanceTable **Tags:** atp.Status=draft
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

No Included Parameters

Included Containers		
Container Name	**Multiplicity**	**Description**
EthSwtAtsInstanceEntry	0..*	Configuration of an Asynchronous Traffic Scheduler configuration in the scope of the PSFP. **Tags:** atp.Status=draft

⌋

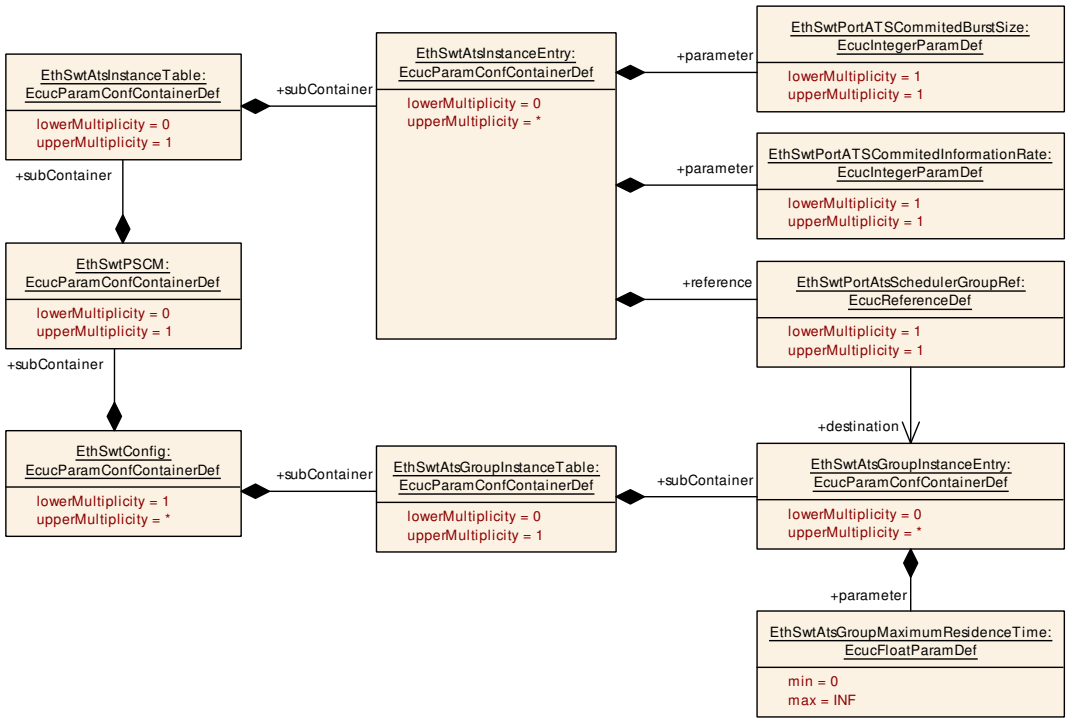

Figure 10.6: EthSwtAtsInstanceTable

10.1.11 EthSwtAtsInstanceEntry

[ECUC_EthSwt_00228] Definition of EcucParamConfContainerDef EthSwtAtsInstanceEntry

Status: DRAFT

⌈

Container Name	EthSwtAtsInstanceEntry	
Parent Container	EthSwtAtsInstanceTable	
Upstream mapping defined	[14, CP_TPS_SystemTemplate]	
Description	Configuration of an Asynchronous Traffic Scheduler configuration in the scope of the PSFP. **Tags:** atp.Status=draft	
Multiplicity	0..*	
Post-Build Multiplicity	Loadable	Selectable
Configuration Parameters		

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
EthSwtPortATSCommitedBurstSize	1	[ECUC_EthSwt_00197]
EthSwtPortATSCommitedInformationRate	1	[ECUC_EthSwt_00198]
EthSwtPortAtsSchedulerGroupRef	1	[ECUC_EthSwt_00196]

No Included Containers

⌋

[ECUC_EthSwt_00197] Definition of EcucIntegerParamDef EthSwtPortATSCommitedBurstSize

Status: DRAFT

⌈

Parameter Name	EthSwtPortATSCommitedBurstSize	
Parent Container	EthSwtAtsInstanceEntry	
Upstream mapping defined	–	
Description	Maximum token capacity of the token bucket. **Tags:** atp.Status=draft	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 18446744073709551615	
Default value	–	
Post-Build Value	Loadable	Selectable

⌋

[ECUC_EthSwt_00198] Definition of EcucIntegerParamDef EthSwtPortATSCommitedInformationRate

Status: DRAFT

⌈

Parameter Name	EthSwtPortATSCommitedInformationRate	
Parent Container	EthSwtAtsInstanceEntry	
Upstream mapping defined	–	
Description	Defines the rate at which the token bucket is refilled with tokens. **Tags:** atp.Status=draft	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 18446744073709551615	
Default value	–	
Post-Build Value	Loadable	Selectable

⌋

[ECUC_EthSwt_00196] Definition of EcucReferenceDef EthSwtPortAtsSchedulerGroupRef

Status: DRAFT

⌈

Parameter Name	EthSwtPortAtsSchedulerGroupRef
Parent Container	EthSwtAtsInstanceEntry
Upstream mapping defined	–

▽

△

Description	Defines to which ATS scheduler group this ATS scheduler belongs to. **Tags:** atp.Status=draft	
Multiplicity	1	
Type	Reference to EthSwtAtsGroupInstanceEntry	
Post-Build Value	Loadable	Selectable

⌋

10.1.12 EthSwtPSFP

[ECUC_EthSwt_00154] Definition of EcucParamConfContainerDef EthSwtPSFP

Status: DRAFT

⌈

Container Name	EthSwtPSFP
Parent Container	EthSwtPSCM
Upstream mapping defined	–
Description	Configuration of Per-stream Filtering and Policing (PSFP). **Tags:** atp.Status=draft
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

No Included Parameters

Included Containers		
Container Name	**Multiplicity**	**Description**
EthSwtFilterMaxSduSizeTable	0..1	EthSwtFilterMaxSduSizeTable represents a table of sdu size values, where each value (table entry) could be referenced by a EthSwtStreamFilterEntry. **Tags:** atp.Status=draft
EthSwtFlowMeteringTable	0..1	EthSwtFlowMeteringTable represents a table of flowmeter configurations, where each flowmeter configuration (table entry) could be referenced by a EthSwtStreamFilterEntry. **Tags:** atp.Status=draft
EthSwtStreamFilterTable	0..1	EthSwtStreamFilterTable represents a table of stream filter configurations, where each stream filter configuration (table entry) could reference a EthSwtFilterMaxSduSizeEntry, EthSwt FlowMeteringEntry and EthSwtStreamGateEntry. **Tags:** atp.Status=draft
EthSwtStreamGateTable	0..1	EthSwtStreamGateTable represents a table of stream gate configurations, where each stream gate configuration (table entry) could be referenced by a EthSwtStreamFilterEntry. **Tags:** atp.Status=draft

⌋

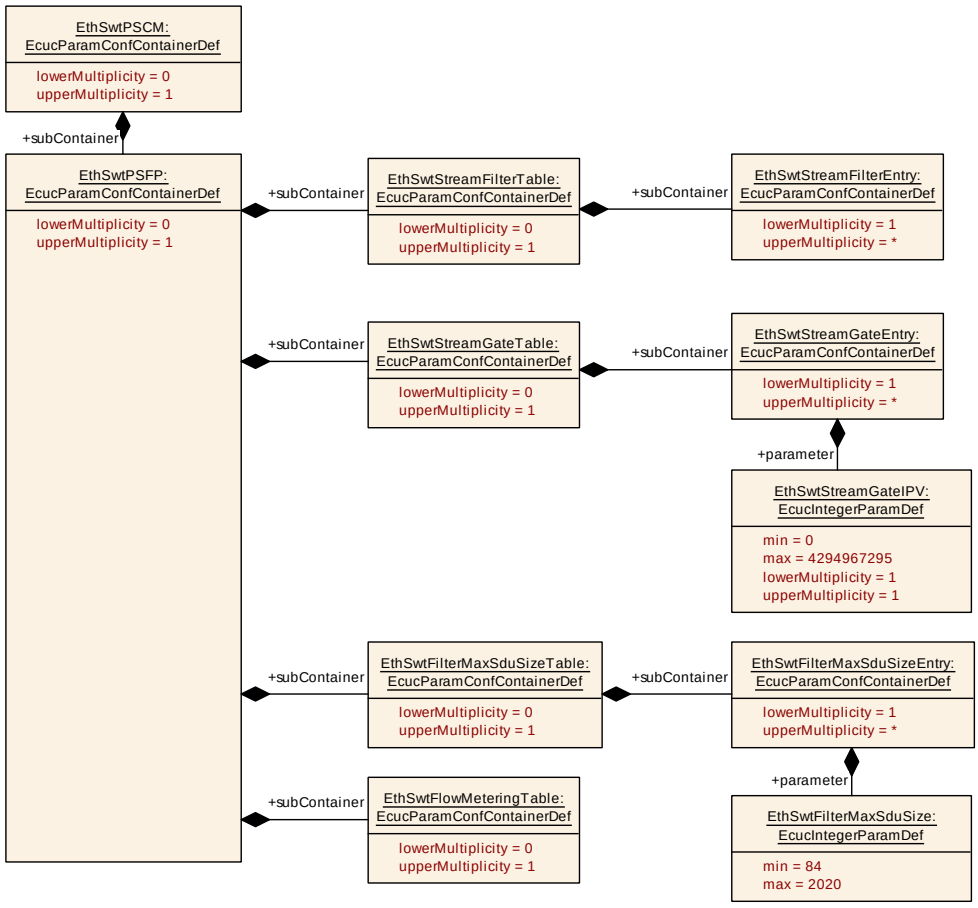

Figure 10.7: EthSwtPSFP

10.1.13 EthSwtFilterMaxSduSizeTable

[ECUC_EthSwt_00222] Definition of EcucParamConfContainerDef EthSwtFilterMaxSduSizeTable

Status: DRAFT

⌈

Container Name	EthSwtFilterMaxSduSizeTable
Parent Container	EthSwtPSFP
Upstream mapping defined	–
Description	EthSwtFilterMaxSduSizeTable represents a table of sdu size values, where each value (table entry) could be referenced by a EthSwtStreamFilterEntry. **Tags:** atp.Status=draft
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

No Included Parameters

Included Containers		
Container Name	**Multiplicity**	**Description**
EthSwtFilterMaxSduSizeEntry	1..*	EthSwtFilterMaxSduSizeEntry defines the maximum SDU size (size of an Ethernet packet) which is acceptable to be processed by the Ethernet switch. The value of EthSwtFilterMaxSduSizeEntry consider the size of the following parts of an Ethernet packet: • Preamble (7 byte) • SFD (start of frame delimiter) (1 byte) • Ethernet frame (Dst MAC,Src MAC, VLAN-tag, TypeField, Payload, CRC Checksum) • Minimum IPG (inter package gap) (12 byte times). **Tags:** atp.Status=draft

⌋

10.1.14 EthSwtFilterMaxSduSizeEntry

[ECUC_EthSwt_00224] Definition of EcucParamConfContainerDef EthSwtFilterMaxSduSizeEntry

Status: DRAFT

⌈

Container Name	EthSwtFilterMaxSduSizeEntry	
Parent Container	EthSwtFilterMaxSduSizeTable	
Upstream mapping defined	[14, CP_TPS_SystemTemplate]	
Description	EthSwtFilterMaxSduSizeEntry defines the maximum SDU size (size of an Ethernet packet) which is acceptable to be processed by the Ethernet switch. The value of EthSwtFilterMaxSduSizeEntry consider the size of the following parts of an Ethernet packet: • Preamble (7 byte) • SFD (start of frame delimiter) (1 byte) • Ethernet frame (Dst MAC,Src MAC, VLAN-tag, TypeField, Payload, CRC Checksum) • Minimum IPG (inter package gap) (12 byte times). **Tags:** atp.Status=draft	
Multiplicity	1..*	
Post-Build Multiplicity	Loadable	Selectable
Configuration Parameters		

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
EthSwtFilterMaxSduSize	1	[ECUC_EthSwt_00225]

No Included Containers

⌋

[ECUC_EthSwt_00225] Definition of EcucIntegerParamDef EthSwtFilterMaxSduSize

Status: DRAFT

⌈

Parameter Name	EthSwtFilterMaxSduSize	
Parent Container	EthSwtFilterMaxSduSizeEntry	
Upstream mapping defined	–	
Description	Max Sdu size. **Tags:** atp.Status=draft	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	84 .. 2020	
Default value	–	
Post-Build Value	Loadable	Selectable

⌋

10.1.15 EthSwtFlowMeteringTable

[ECUC_EthSwt_00219] Definition of EcucParamConfContainerDef EthSwtFlowMeteringTable

Status: DRAFT

⌈

Container Name	EthSwtFlowMeteringTable
Parent Container	EthSwtPSFP
Upstream mapping defined	–
Description	EthSwtFlowMeteringTable represents a table of flowmeter configurations, where each flowmeter configuration (table entry) could be referenced by a EthSwtStreamFilterEntry. **Tags:** atp.Status=draft
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

No Included Parameters

Included Containers		
Container Name	**Multiplicity**	**Description**
EthSwtFlowMeteringEntry	1..*	Configuration of a flow metering. **Tags:** atp.Status=draft

⌋

10.1.16 EthSwtFlowMeteringEntry

[ECUC_EthSwt_00157] Definition of EcucParamConfContainerDef EthSwtFlowMeteringEntry

Status: DRAFT

⌈

Container Name	EthSwtFlowMeteringEntry	
Parent Container	EthSwtFlowMeteringTable	
Upstream mapping defined	[14, CP_TPS_SystemTemplate]	
Description	Configuration of a flow metering. **Tags:** atp.Status=draft	
Multiplicity	1..*	
Post-Build Multiplicity	Loadable	Selectable
Configuration Parameters		

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
EthSwtFlowMeterCF	1	[ECUC_EthSwt_00162]
EthSwtFlowMeteringCBS	1	[ECUC_EthSwt_00159]
EthSwtFlowMeteringCIR	1	[ECUC_EthSwt_00158]
EthSwtFlowMeteringColorMode	1	[ECUC_EthSwt_00163]
EthSwtFlowMeteringEBS	1	[ECUC_EthSwt_00161]
EthSwtFlowMeteringEIR	1	[ECUC_EthSwt_00160]

No Included Containers

⌋

[ECUC_EthSwt_00162] Definition of EcucBooleanParamDef EthSwtFlowMeterCF

Status: DRAFT

⌈

Parameter Name	EthSwtFlowMeterCF	
Parent Container	EthSwtFlowMeteringEntry	
Upstream mapping defined	–	
Description	Coupling Flag that defines if unused "green" tokens in the first bucket are transferred to the second bucket as "yellow" tokens. Note: this parameter maps to IEEE802.1Q parameter "ieee8021PSFPFlowMeterCF". **Tags:** atp.Status=draft	
Multiplicity	1	
Type	EcucBooleanParamDef	
Default value	–	
Post-Build Value	Loadable	Selectable

⌋

[ECUC_EthSwt_00159] Definition of EcucIntegerParamDef EthSwtFlowMetering CBS

Status: DRAFT

⌈

Parameter Name	EthSwtFlowMeteringCBS	
Parent Container	EthSwtFlowMeteringEntry	
Upstream mapping defined	–	
Description	Committed Burst Size (accepted burst size in green token bucket). Note: this parameter maps to IEEE802.1Q parameter "ieee8021PSFPFlowMeterCBS". **Tags:** atp.Status=draft	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 255	
Default value	–	
Post-Build Value	Loadable	Selectable

⌋

[ECUC_EthSwt_00158] Definition of EcucIntegerParamDef EthSwtFlowMetering CIR

Status: DRAFT

⌈

Parameter Name	EthSwtFlowMeteringCIR	
Parent Container	EthSwtFlowMeteringEntry	
Upstream mapping defined	–	
Description	Committed Information Rate (accepted rate in green token bucket) in bits per second. Note: this parameter maps to IEEE802.1Q parameter "ieee8021PSFPFlowMeterCIR". **Tags:** atp.Status=draft	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 4294967295	
Default value	–	
Post-Build Value	Loadable	Selectable

⌋

[ECUC_EthSwt_00163] Definition of EcucEnumerationParamDef EthSwtFlowMeteringColorMode

Status: DRAFT

⌈

Parameter Name	EthSwtFlowMeteringColorMode	
Parent Container	EthSwtFlowMeteringEntry	
Upstream mapping defined	–	

▽

△

Description	Parameter that defines if color-aware or color-blind mode is used. The mode indicates if a color that might be assigned at ingress is used to chose the bucket from which to take tokens; only green and yellow can be assigned; basically, in color-blind mode, all frames are treated like green frames. Note: this parameter maps to IEEE802.1Q parameter "ieee8021PSFPFlowMeterCM". **Tags:** atp.Status=draft	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	ETHSWT_COLOR_AWARE	color aware color mode. **Tags:** atp.Status=draft
	ETHSWT_COLOR_BLIND	color blind color mode. **Tags:** atp.Status=draft
Post-Build Value	Loadable	Selectable

⌋

[ECUC_EthSwt_00161] Definition of EcucIntegerParamDef EthSwtFlowMeteringEBS

Status: DRAFT

⌈

Parameter Name	EthSwtFlowMeteringEBS	
Parent Container	EthSwtFlowMeteringEntry	
Upstream mapping defined	–	
Description	Excess burst size (accepted burst size in yellow token bucket). Note: this parameter maps to IEEE802.1Q parameter "ieee8021PSFPFlowMeterEBS". **Tags:** atp.Status=draft	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 255	
Default value	–	
Post-Build Value	Loadable	Selectable

⌋

[ECUC_EthSwt_00160] Definition of EcucIntegerParamDef EthSwtFlowMeteringEIR

Status: DRAFT

⌈

Parameter Name	EthSwtFlowMeteringEIR	
Parent Container	EthSwtFlowMeteringEntry	
Upstream mapping defined	–	
Description	Excess Information Rate (accepted rate in yellow token bucket) in bits per second. Note: this parameter maps to IEEE802.1Q parameter "ieee8021PSFPFlowMeterEIR". **Tags:** atp.Status=draft	

▽

△

Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 4294967295	
Default value	–	
Post-Build Value	Loadable	Selectable

⌋

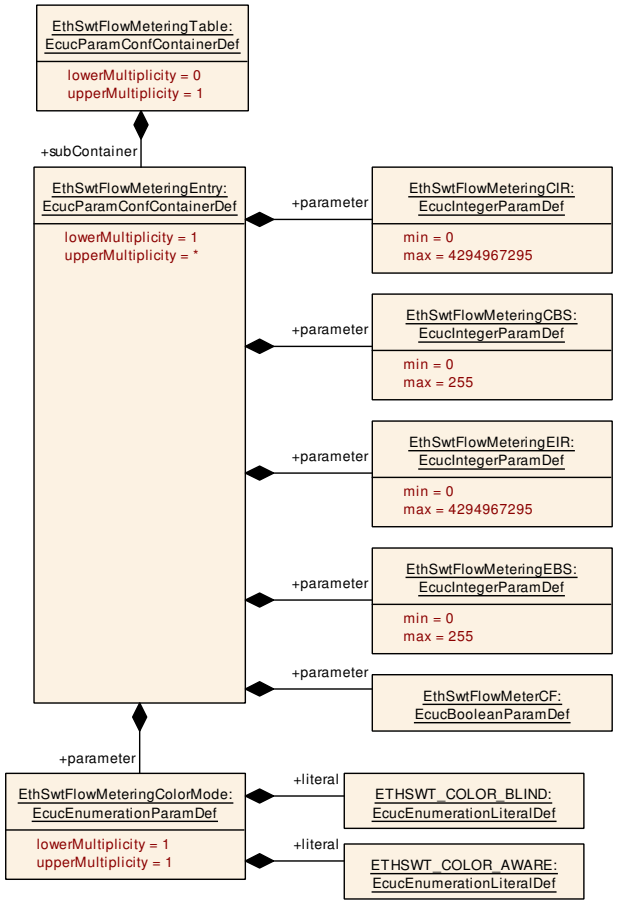

Figure 10.8: `EthSwtFlowMeteringEntry`

10.1.17 EthSwtStreamFilterTable

[ECUC_EthSwt_00214] Definition of EcucParamConfContainerDef EthSwtStream FilterTable

Status: DRAFT

⌈

Container Name	EthSwtStreamFilterTable
Parent Container	EthSwtPSFP
Upstream mapping defined	–
Description	EthSwtStreamFilterTable represents a table of stream filter configurations, where each stream filter configuration (table entry) could reference a EthSwtFilterMaxSduSize Entry, EthSwtFlowMeteringEntry and EthSwtStreamGateEntry. **Tags:** atp.Status=draft
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

No Included Parameters

Included Containers		
Container Name	Multiplicity	Description
EthSwtStreamFilterEntry	1..*	This container represents a stream filter, where EthSwtStream FilterPriority and EthSwtAssignedStreamHandle are used to detect a matching Ethernet frame. **Tags:** atp.Status=draft

⌋

10.1.18 EthSwtStreamFilterEntry

[ECUC_EthSwt_00217] Definition of EcucParamConfContainerDef EthSwtStream FilterEntry

Status: DRAFT

⌈

Container Name	EthSwtStreamFilterEntry	
Parent Container	EthSwtStreamFilterTable	
Upstream mapping defined	[14, CP_TPS_SystemTemplate]	
Description	This container represents a stream filter, where EthSwtStreamFilterPriority and EthSwt AssignedStreamHandle are used to detect a matching Ethernet frame. **Tags:** atp.Status=draft	
Multiplicity	1..*	
Post-Build Multiplicity	Loadable	Selectable
Configuration Parameters		

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
EthSwtStreamFilterEntryPosition	1	[ECUC_EthSwt_00216]
EthSwtStreamFilterPriority	1	[ECUC_EthSwt_00215]
EthSwtAtsInstanceEntryRef	0..1	[ECUC_EthSwt_00227]
EthSwtFilterMaxSduSizeRef	0..1	[ECUC_EthSwt_00223]

▽

△

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
EthSwtFlowMeteringEntryRef	0..1	[ECUC_EthSwt_00221]
EthSwtStreamGateEntryRef	0..1	[ECUC_EthSwt_00213]

Included Containers		
Container Name	**Multiplicity**	**Description**
EthSwtAssignedStreamHandle	1	This container represents an assigned stream handle id of this stream filter, which is used to detect a matching Ethernet frame. The EthSwtAssignedStreamHandle could represent a list of assigned stream handle ids. Additionally it could set a wildcard flag (EthSwtStreamHandleId Wildcard), where any assigned stream handle id carried by an Ethernet frame would match to this stream filter. **Tags:** atp.Status=draft

⌋

[ECUC_EthSwt_00216] Definition of EcucIntegerParamDef EthSwtStreamFilterEntryPosition

Status: DRAFT

⌈

Parameter Name	EthSwtStreamFilterEntryPosition	
Parent Container	EthSwtStreamFilterEntry	
Upstream mapping defined	–	
Description	Specifies the position as unique ID within an ordered list of EthSwtStreamFilterEntrys. The ordered list shall start with 0 and continue as linear list with no gaps. Note: The list is processed in ascending order. The instance of EthSwtStreamFilter Entry with position 0 is processed first. **Tags:** atp.Status=draft	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 65535	
Default value	–	
Post-Build Value	Loadable	Selectable

⌋

[ECUC_EthSwt_00215] Definition of EcucIntegerParamDef EthSwtStreamFilterPriority

Status: DRAFT

⌈

Parameter Name	EthSwtStreamFilterPriority
Parent Container	EthSwtStreamFilterEntry
Upstream mapping defined	–

▽

△

Description	This parameter represents an assigned priority of this stream filter, which is used to detect a matching Ethernet frame. **Tags:** atp.Status=draft	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 7	
Default value	–	
Post-Build Value	Loadable	Selectable

⌋

[ECUC_EthSwt_00227] Definition of EcucReferenceDef EthSwtAtsInstanceEntryRef

Status: DRAFT

⌈

Parameter Name	EthSwtAtsInstanceEntryRef	
Parent Container	EthSwtStreamFilterEntry	
Upstream mapping defined	–	
Description	Reference to an entry of an ATS table, where the entry represents a configuration for asynchronous traffic shaping. **Tags:** atp.Status=draft	
Multiplicity	0..1	
Type	Reference to EthSwtAtsInstanceEntry	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

⌋

[ECUC_EthSwt_00223] Definition of EcucReferenceDef EthSwtFilterMaxSduSizeRef

Status: DRAFT

⌈

Parameter Name	EthSwtFilterMaxSduSizeRef	
Parent Container	EthSwtStreamFilterEntry	
Upstream mapping defined	–	
Description	Reference to an entry of a max-sdu-size table, where the entry represents a particular value. **Tags:** atp.Status=draft	
Multiplicity	0..1	
Type	Reference to EthSwtFilterMaxSduSizeEntry	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

⌋

[ECUC_EthSwt_00221] Definition of EcucReferenceDef EthSwtFlowMeteringEntryRef

Status: DRAFT

⌈

Parameter Name	EthSwtFlowMeteringEntryRef	
Parent Container	EthSwtStreamFilterEntry	
Upstream mapping defined	–	
Description	Reference to an entry of a flow metering table, where the entry represents a configuration for flow metering. **Tags:** atp.Status=draft	
Multiplicity	0..1	
Type	Reference to EthSwtFlowMeteringEntry	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

⌋

[ECUC_EthSwt_00213] Definition of EcucReferenceDef EthSwtStreamGateEntryRef

Status: DRAFT

⌈

Parameter Name	EthSwtStreamGateEntryRef	
Parent Container	EthSwtStreamFilterEntry	
Upstream mapping defined	–	
Description	Reference to an entry of a gate table, where the entry represents a configuration for a gate. **Tags:** atp.Status=draft	
Multiplicity	0..1	
Type	Reference to EthSwtStreamGateEntry	
Post-Build Value	Loadable	Selectable

⌋

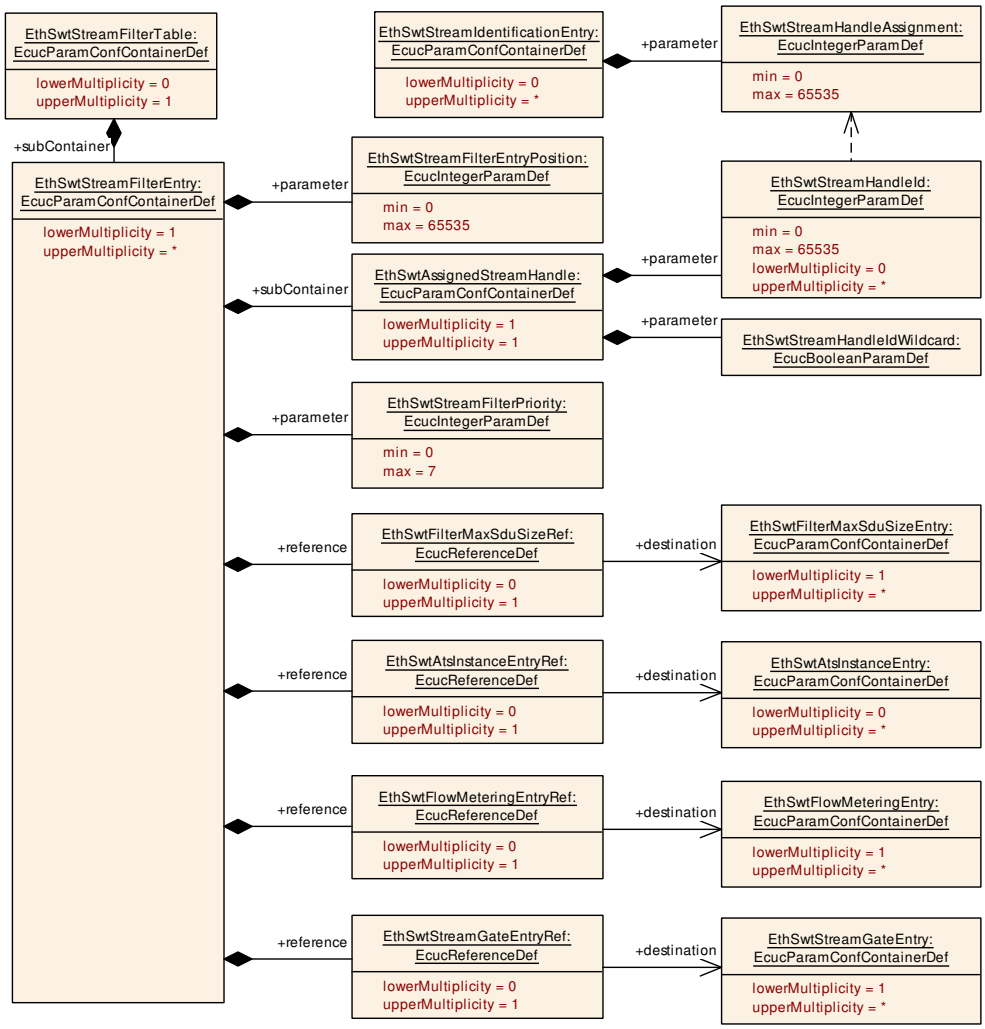

Figure 10.9: `EthSwtStreamFilterEntry`

10.1.19 EthSwtAssignedStreamHandle

[ECUC_EthSwt_00231] Definition of EcucParamConfContainerDef EthSwtAssignedStreamHandle

Status: DRAFT

⌈

Container Name	EthSwtAssignedStreamHandle
Parent Container	EthSwtStreamFilterEntry
Upstream mapping defined	–

▽

△

Description	This container represents an assigned stream handle id of this stream filter, which is used to detect a matching Ethernet frame. The EthSwtAssignedStreamHandle could represent a list of assigned stream handle ids. Additionally it could set a wildcard flag (EthSwtStreamHandleIdWildcard), where any assigned stream handle id carried by an Ethernet frame would match to this stream filter. **Tags:** atp.Status=draft
Multiplicity	1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
EthSwtStreamHandleId	0..*	[ECUC_EthSwt_00210]
EthSwtStreamHandleIdWildcard	1	[ECUC_EthSwt_00209]

No Included Containers

⌋

[ECUC_EthSwt_00210] Definition of EcucIntegerParamDef EthSwtStreamHandleId

Status: DRAFT

⌈

Parameter Name	EthSwtStreamHandleId	
Parent Container	EthSwtAssignedStreamHandle	
Upstream mapping defined	–	
Description	Assigned stream handle id of this stream filter, which is used for evaluation of a matching Ethernet frame. **Tags:** atp.Status=draft	
Multiplicity	0..*	
Type	EcucIntegerParamDef	
Range	0 .. 65535	
Default value	–	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

⌋

[ECUC_EthSwt_00209] Definition of EcucBooleanParamDef EthSwtStreamHandleIdWildcard

Status: DRAFT

⌈

Parameter Name	EthSwtStreamHandleIdWildcard	
Parent Container	EthSwtAssignedStreamHandle	
Upstream mapping defined	–	
Description	Defines whether this EthSwtAssignedStreamHandle includes the wildcard. **Tags:** atp.Status=draft	
Multiplicity	1	
Type	EcucBooleanParamDef	
Default value	–	
Post-Build Value	Loadable	Selectable

⌋

10.1.20 EthSwtStreamGateTable

[ECUC_EthSwt_00212] Definition of EcucParamConfContainerDef EthSwtStreamGateTable

Status: DRAFT

⌈

Container Name	EthSwtStreamGateTable
Parent Container	EthSwtPSFP
Upstream mapping defined	–
Description	EthSwtStreamGateTable represents a table of stream gate configurations, where each stream gate configuration (table entry) could be referenced by a EthSwtStreamFilter Entry. **Tags:** atp.Status=draft
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

No Included Parameters

Included Containers		
Container Name	**Multiplicity**	**Description**
EthSwtStreamGateEntry	1..*	Configuration of a stream gate. **Tags:** atp.Status=draft

⌋

10.1.21 EthSwtStreamGateEntry

[ECUC_EthSwt_00155] Definition of EcucParamConfContainerDef EthSwtStreamGateEntry

Status: DRAFT

⌈

Container Name	EthSwtStreamGateEntry	
Parent Container	EthSwtStreamGateTable	
Upstream mapping defined	[14, CP_TPS_SystemTemplate]	
Description	Configuration of a stream gate. **Tags:** atp.Status=draft	
Multiplicity	1..*	
Post-Build Multiplicity	Loadable	Selectable
Configuration Parameters		

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
EthSwtStreamGateIPV	1	[ECUC_EthSwt_00156]

No Included Containers

⌋

[ECUC_EthSwt_00156] Definition of EcucIntegerParamDef EthSwtStreamGateIPV

Status: DRAFT

⌈

Parameter Name	EthSwtStreamGateIPV	
Parent Container	EthSwtStreamGateEntry	
Upstream mapping defined	–	
Description	Internal Priority Value (IPV), a priority value that determines the assigned traffic class. Note: The upper value is limited by the configured used internal priority value of the this Ethernet switch (EthSwtUsedInternalPriorityUpperValue). The remaining value range shall be ignored. **Tags:** atp.Status=draft	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 4294967295	
Default value	–	
Post-Build Value	Loadable	Selectable

⌋

10.1.22 EthSwtPort

[ECUC_EthSwt_00005] Definition of EcucParamConfContainerDef EthSwtPort ⌈

Container Name	EthSwtPort	
Parent Container	EthSwtConfig	
Upstream mapping defined	[14, CP_TPS_SystemTemplate]	
Description	Configuration of one Ethernet Switch Port.	
Multiplicity	1..*	
Post-Build Multiplicity	Loadable	Selectable
Configuration Parameters		

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
EthSwtFramePreemptionEnable	1	[ECUC_EthSwt_00254]
EthSwtPortIdx	1	[ECUC_EthSwt_00013]
EthSwtPortInterPacketGap	0..1	[ECUC_EthSwt_00238]
EthSwtPortMacLayerSpeed	0..1	[ECUC_EthSwt_00114]
EthSwtPortMacLayerSubType	0..1	[ECUC_EthSwt_00113]
EthSwtPortMacLayerType	0..1	[ECUC_EthSwt_00072]
EthSwtPortMacSecBypassDestinationMacAddress	0..255	[ECUC_EthSwt_00265]
EthSwtPortMacSecBypassEtherType	0..255	[ECUC_EthSwt_00262]
EthSwtPortMacSecBypassVlan	0..255	[ECUC_EthSwt_00263]
EthSwtPortMacSecEnabled	0..1	[ECUC_EthSwt_00264]
EthSwtPortPhysicalLayerType	0..1	[ECUC_EthSwt_00054]
EthSwtPortRole	0..1	[ECUC_EthSwt_00101]
EthSwtPortTimeStampSupport	1	[ECUC_EthSwt_00112]
EthSwtPortTrcvRef	0..1	[ECUC_EthSwt_00041]

Included Containers		
Container Name	**Multiplicity**	**Description**
EthSwtPortEgress	1	Configuration of one Ethernet Switch Port Egress behavior.
EthSwtPortIngress	1	Configuration of one Ethernet Switch Port ingress behavior.

⌋

[ECUC_EthSwt_00254] Definition of EcucBooleanParamDef EthSwtFramePreemptionEnable

Status: DRAFT

⌈

Parameter Name	EthSwtFramePreemptionEnable
Parent Container	EthSwtPort
Upstream mapping defined	[14, CP_TPS_SystemTemplate]

▽

△

Description	Configures whether frame preemption for this EthSwtPort is enabled. If the Ethernet switch hardware supports frame preemption and this parameter is set TRUE, then frame preemption for the corresponding EthSwtPort is enabled. If for some traffic class(es) EthSwtTrafficClassToPreemptionStatusAssignment at the egress port queue is additionally configured to ETHSWT_TRAFFIC_CLASS_PREEMPTABLE, then frame preemption for the respective traffic class(es) at the correponding EthSwtEgress Port is possible. **Tags:** atp.Status=draft Dependency: If an Ethernet switch hardware does not support frame preemption, then this parameter shall be set to false.	
Multiplicity	1	
Type	EcucBooleanParamDef	
Default value	false	
Post-Build Value	Loadable	Selectable

⌋

[ECUC_EthSwt_00013] Definition of EcucIntegerParamDef EthSwtPortIdx ⌈

Parameter Name	EthSwtPortIdx	
Parent Container	EthSwtPort	
Upstream mapping defined	–	
Description	Specifies the instance ID of the configured Ethernet Switch Port.	
Multiplicity	1	
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)	
Range	0 .. 255	
Default value	–	
With Auto	true	
Pre-Compile only	true	

⌋

[ECUC_EthSwt_00238] Definition of EcucIntegerParamDef EthSwtPortInterPacketGap ⌈

Parameter Name	EthSwtPortInterPacketGap	
Parent Container	EthSwtPort	
Upstream mapping defined	–	
Description	This parameter defines the transmit Inter-Packet Gap (IPG) (also called interframe gap (IFG)) between transmitted Ethernet packets in "byte times". This parameter can only exist if the hardware and driver support setting the IPG.	
Multiplicity	0..1	
Type	EcucIntegerParamDef	
Range	0 .. 10000	
Default value	12	
Pre-Compile only	true	

⌋

[ECUC_EthSwt_00114] Definition of EcucEnumerationParamDef EthSwtPortMacLayerSpeed ⌈

Parameter Name	EthSwtPortMacLayerSpeed	
Parent Container	EthSwtPort	
Upstream mapping defined	–	
Description	Defines the baud rate of the MAC layer.	
Multiplicity	0..1	
Type	EcucEnumerationParamDef	
Range	ETH_MAC_LAYER_SPEED_100M	100 MBit/s
	ETH_MAC_LAYER_SPEED_10G	10 GBit/s
	ETH_MAC_LAYER_SPEED_10M	10 MBit/s
	ETH_MAC_LAYER_SPEED_1G	1 GBit/s
	ETH_MAC_LAYER_SPEED_2500M	2500 MBit/s (2.5 GBit/s)
	ETH_MAC_LAYER_SPEED_5G	5 GBit/s
Post-Build Multiplicity	-	Selectable
Post-Build Value	-	Selectable

⌋

[ECUC_EthSwt_00113] Definition of EcucEnumerationParamDef EthSwtPortMacLayerSubType ⌈

Parameter Name	EthSwtPortMacLayerSubType	
Parent Container	EthSwtPort	
Upstream mapping defined	–	
Description	Defines the MAC layer subtype of this EthSwtPort.	
Multiplicity	0..1	
Type	EcucEnumerationParamDef	
Range	REDUCED	Reduced media-independent interface (RMII)
	REVERSED	Reversed media-independent interface (to provide direct connection between two Ethernet MACs)
	SERIAL	Low-power and low pin-count serial 8b/10b-coded media-independent interface
	STANDARD	Standard media-independent interface
	UNIVERSAL_SERIAL	Universal low-power and low pin-count serial 8b/10b-coded media-independent interface
Post-Build Multiplicity	-	Selectable
Post-Build Value	-	Selectable

⌋

[ECUC_EthSwt_00072] Definition of EcucEnumerationParamDef EthSwtPortMacLayerType ⌈

Parameter Name	EthSwtPortMacLayerType
Parent Container	EthSwtPort
Upstream mapping defined	[14, CP_TPS_SystemTemplate]

▽

△

Description	Defines the MAC layer type of this EthSwtPort.	
Multiplicity	0..1	
Type	EcucEnumerationParamDef	
Range	ETHSWT_PORT_MAC_LAYER_TYPE_XGMII	MAC layer interface (data) bandwith class 1Gbit/s (e.g. GMII, RGMII, SGMII, RvGMII, USGMII)
	ETHSWT_PORT_MAC_LAYER_TYPE_XMII	MAC layer interface (data) bandwith class 100 Mbit/s (e.g. MII, RMII, RvMII, SMII, RvMII)
	ETHSWT_PORT_MAC_LAYER_TYPE_XXGMII	MAC layer interface (data) bandwith class 10 Gbit/s
Post-Build Multiplicity	-	Selectable
Post-Build Value	-	Selectable

⌋

[ECUC_EthSwt_00265] Definition of EcucStringParamDef EthSwtPortMacSecBypassDestinationMacAddress ⌈

Parameter Name	EthSwtPortMacSecBypassDestinationMacAddress	
Parent Container	EthSwtPort	
Upstream mapping defined	[14, CP_TPS_SystemTemplate]	
Description	Bypassed destination MAC address (48-bit physical address). Ethernet frames, which carrying the configured destination MAC address, shall bypass the MACsec protection handling.	
Multiplicity	0..255	
Type	EcucStringParamDef	
Default value	–	
Length	17-17	
Regular Expression	([0-9a-fA-F]{2}:){5}[0-9a-fA-F]{2}	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

⌋

[ECUC_EthSwt_00262] Definition of EcucIntegerParamDef EthSwtPortMacSecBypassEtherType ⌈

Parameter Name	EthSwtPortMacSecBypassEtherType	
Parent Container	EthSwtPort	
Upstream mapping defined	[14, CP_TPS_SystemTemplate]	
Description	Bypassed EtherType. Ethernet frames, which carrying the configured EtherType, shall bypass the MACsec protection handling.	
Multiplicity	0..255	
Type	EcucIntegerParamDef	
Range	0 .. 65535	
Default value	–	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

⌋

[ECUC_EthSwt_00263] Definition of EcucIntegerParamDef EthSwtPortMacSecBypassVlan ⌈

Parameter Name	EthSwtPortMacSecBypassVlan	
Parent Container	EthSwtPort	
Upstream mapping defined	[14, CP_TPS_SystemTemplate]	
Description	Bypassed VLAN-ID. Ethernet frames, which carrying the configured VLAN-ID, shall bypass the MACsec protection handling. Please note: VLAN-ID 0 is interpreted as no VLAN, i.e. untagged Ethernet frames bypass the MACsec protection handling.	
Multiplicity	0..255	
Type	EcucIntegerParamDef	
Range	0 .. 4094	
Default value	–	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

⌋

[ECUC_EthSwt_00264] Definition of EcucBooleanParamDef EthSwtPortMacSecEnabled ⌈

Parameter Name	EthSwtPortMacSecEnabled
Parent Container	EthSwtPort
Upstream mapping defined	–
Description	Specifies if MACsec according to IEEE802.1AE (MACsec protocol) and IEEE802.1 X(MACsec Key Agreement protocol) is supported by the used Ethernet hardware (PHY) and if the functionality (MACsec protection) is enabled. If the parameter is configured, MACsec is supported according to IEEE802.1AE and IEEE802.1X by the used Ethernet hardware (PHY). If the parameter is set to TRUE, the MACsec functionality according to IEEE802.1AE and IEEE802.1X is enabled. If the parameter is set to FALSE, the MACsec functionality according to according to IEEE802.1AE and IEEE802.1X is disabled. Note: Disabling of the MACsec functionality is used if one of the connected link partner uses Ethernet hardware (PHY or Ethernet switch port) which do not support the MACsec functionality. This may be needed for legacy scenarios, where ECUs are taken over from previous car lines.
Multiplicity	0..1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_EthSwt_00054] Definition of EcucEnumerationParamDef EthSwtPortPhysicalLayerType ⌈

Parameter Name	EthSwtPortPhysicalLayerType
Parent Container	EthSwtPort
Upstream mapping defined	[14, CP_TPS_SystemTemplate]
Description	Defines the physical layer type of this EthSwtPort. Dependency: If a EthSwtPort has an EthSwtPortPhysicalLayerType then EthSwtPort shall reference an EthTrcv.
Multiplicity	0..1

▽

△

Type	EcucEnumerationParamDef	
Range	ETHSWT_PORT_10000BASE_T1	physical layer interface 10GBASE-T1 (10Gbit/s, 1pair). Used for automotive.
	ETHSWT_PORT_1000BASE_T	physical layer interface 1000BASE-T (1Gbit/s, 4 pairs). Used for consumer electronic.
	ETHSWT_PORT_1000BASE_T1	physical layer interface 1000BASE-T1 (1Gbit/s, 1 pair). Used for automotive.
	ETHSWT_PORT_100BASE_T1	physical layer interface 100BASE-T1 (100Mbit/s, 1 pair). Used for automotive.
	ETHSWT_PORT_100BASE_TX	physical layer interface 100BASE-TX (100Mbit/s, 2 pairs). Used for consumer electronic.
	ETHSWT_PORT_10BASE_T1S	physical layer interface 10BASE-T1S (10Mbit/s, 1 pair). Used for automotive.
	ETHSWT_PORT_2500BASE_T1	physical layer interface 2.5GBASE-T1 (2.5Gbit/s, 1pair). Used for automotive.
	ETHSWT_PORT_5000BASE_T1	physical layer interface 5GBASE-T1 (5Gbit/s, 1pair). Used for automotive.
Post-Build Multiplicity	-	Selectable
Post-Build Value	-	Selectable

⌋

[ECUC_EthSwt_00101] Definition of EcucEnumerationParamDef EthSwtPortRole

⌈

Parameter Name	EthSwtPortRole	
Parent Container	EthSwtPort	
Upstream mapping defined	[14, CP_TPS_SystemTemplate]	
Description	Set a special role of the Ethernet switch port. It is either a host port or a up link port. If not configured it is a standard port. Dependency: One Ethernet switch shall have either exactly one host port or at least one up link port. In case of having a host port also multiple up link port can exist. A master switch shall be connected by one host port with the host ecu. A slave switch shall be connected to a master switch by one up link port.	
Multiplicity	0..1	
Type	EcucEnumerationParamDef	
Range	ETHSWT_HOST_PORT	The hostPort is connected to an ECU (host ecu). The host ECU controls the connected Coupling Element (e.g. Ethernet switch).
	ETHSWT_UP_LINK_PORT	A CouplingPort can be connected to another CouplingPort of a CouplingElement located on the same ECU (CouplingElement.ecuInstance) using the CouplingPortConnection. This is used to model a cascaded switch.
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

⌋

[ECUC_EthSwt_00112] Definition of EcucBooleanParamDef EthSwtPortTimeStampSupport ⌈

Parameter Name	EthSwtPortTimeStampSupport	
Parent Container	EthSwtPort	
Upstream mapping defined	–	
Description	Enables/Disables the Switch-port specific timestamping. Dependency: EthSwtPortTimeStampSupport can only be set to TRUE, * if (EthSwt ClockSynchronizationSupport is FALSE) OR * if ((EthSwtClockSynchronizationSupport is TRUE) AND (EthSwtPortRole is NOT ETHSWT_UP_LINK_PORT))	
Multiplicity	1	
Type	EcucBooleanParamDef	
Default value	–	
Post-Build Value	Loadable	Selectable

⌋

[ECUC_EthSwt_00041] Definition of EcucReferenceDef EthSwtPortTrcvRef ⌈

Parameter Name	EthSwtPortTrcvRef	
Parent Container	EthSwtPort	
Upstream mapping defined	[14, CP_TPS_SystemTemplate]	
Description	Reference to the Ethernet transceiver driver this EthSwtPort is connected with. Dependency: If EthSwtPortPhysicalLayerType is defined, then EthSwtPortTrcvRef holds the reference to the corresponding EthTrcv.	
Multiplicity	0..1	
Type	Symbolic name reference to EthTrcvConfig	
Post-Build Multiplicity	-	Selectable
Post-Build Value	Loadable	Selectable

⌋

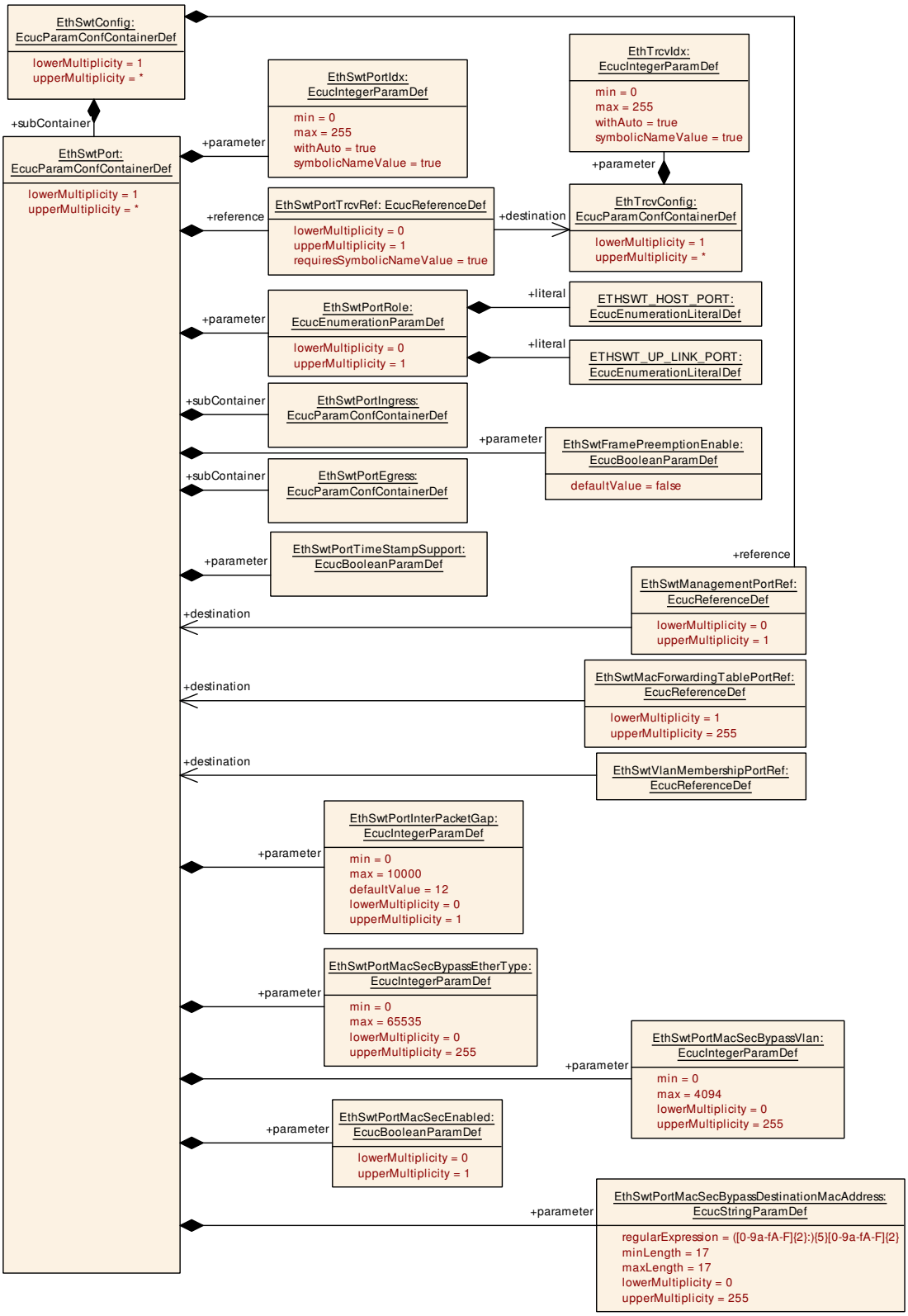

Figure 10.10: `EthSwtPort` (1/2)

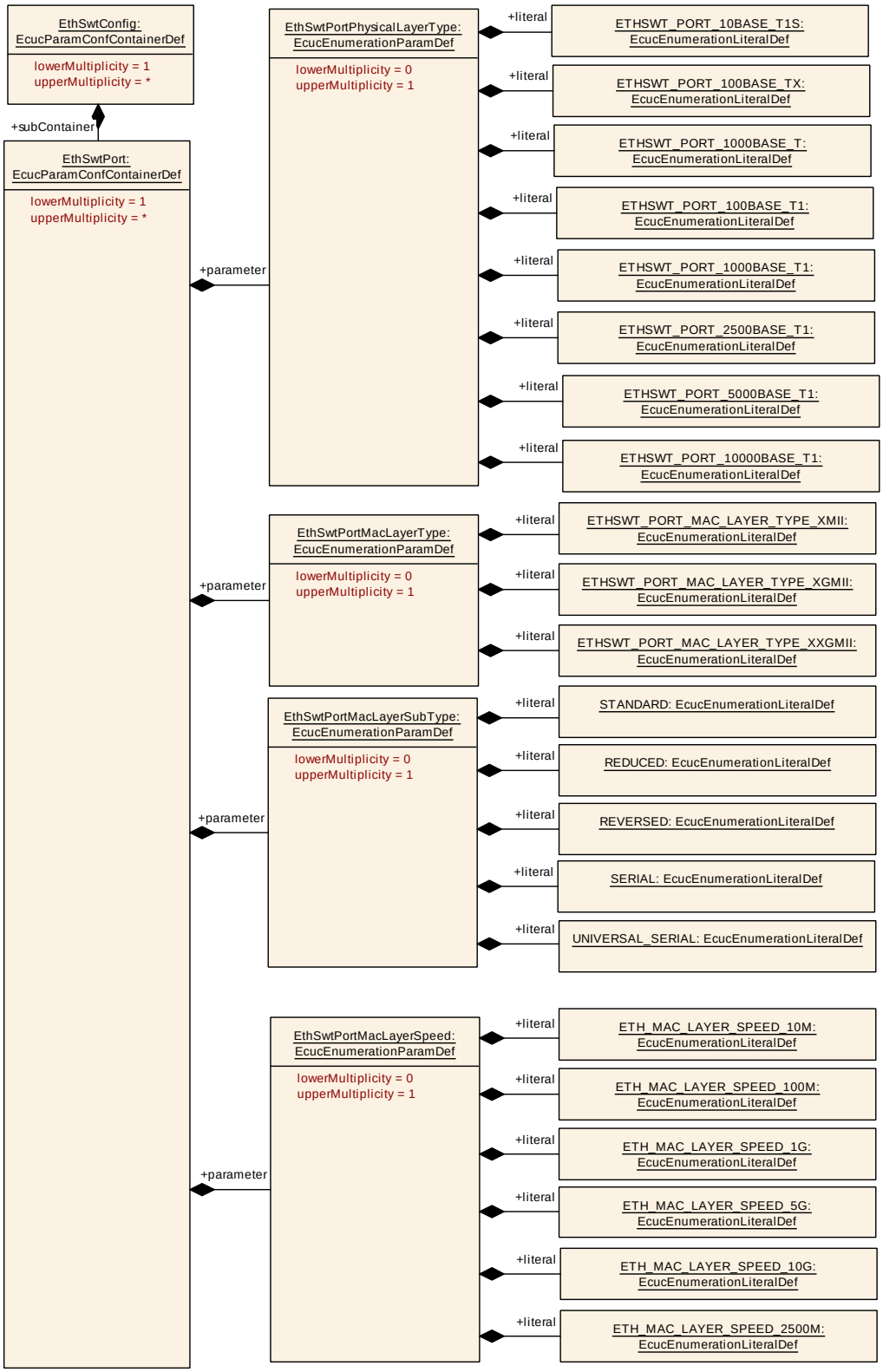

Figure 10.11: `EthSwtPort` (2/2)

Please note that the functional behavior of the ingress and egress port of a switch is implemented in hardware in the switch devices (see [13]). Thus, the configuration of

EthSwtPort and described in the following has to be written to the switch device or is related to the switch configuration.

10.1.23 EthSwtPortEgress

[ECUC_EthSwt_00007] Definition of EcucParamConfContainerDef EthSwtPortEgress ⌈

Container Name	EthSwtPortEgress
Parent Container	EthSwtPort
Upstream mapping defined	–
Description	Configuration of one Ethernet Switch Port Egress behavior.
Multiplicity	1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
EthSwtPortDefaultTrafficClass	1	[ECUC_EthSwt_00247]
EthSwtPortEgressLastSchedulerRef	1	[ECUC_EthSwt_00008]

Included Containers		
Container Name	**Multiplicity**	**Description**
EthSwtPortEgressScheduler	1..*	Represents a Scheduler in the egress port.
EthSwtPortPriorityToTrafficClass Assignment	1..65535	Defines a priority based traffic class assignment. All Ethernet frames with a specific priority (EthSwtPortPriorityToTrafficClass AssignmentPriority) arriving at the egress side within the forwarding process, shall be assigned to the corresponding traffic class (EthSwtPortPriorityToTrafficClassAssignmentTrafficClass). **Tags:** atp.Status=draft
EthSwtPortQueue	1..65535	Represents a Queue at the egress port. **Tags:** atp.Status=draft

⌋

[ECUC_EthSwt_00247] Definition of EcucIntegerParamDef EthSwtPortDefaultTrafficClass

Status: DRAFT

⌈

Parameter Name	EthSwtPortDefaultTrafficClass
Parent Container	EthSwtPortEgress
Upstream mapping defined	[14, CP_TPS_SystemTemplate]
Description	Represents the default traffic class assignment. All Ethernet frames, where the priority associated with this Ethernet frame is not available in a EthSwtPortTrafficClass Assignment of this egress port, are assigned to the default traffic class. **Tags:** atp.Status=draft

▽

△

Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 65535	
Default value	1	
Post-Build Value	Loadable	Selectable

⌋

[ECUC_EthSwt_00008] Definition of EcucReferenceDef EthSwtPortEgressLastSchedulerRef ⌈

Parameter Name	EthSwtPortEgressLastSchedulerRef	
Parent Container	EthSwtPortEgress	
Upstream mapping defined	[14, CP_TPS_SystemTemplate]	
Description	Reference to the port scheduler which is the last in the egress port structure.	
Multiplicity	1	
Type	Reference to EthSwtPortEgressScheduler	
Post-Build Value	Loadable	Selectable

⌋

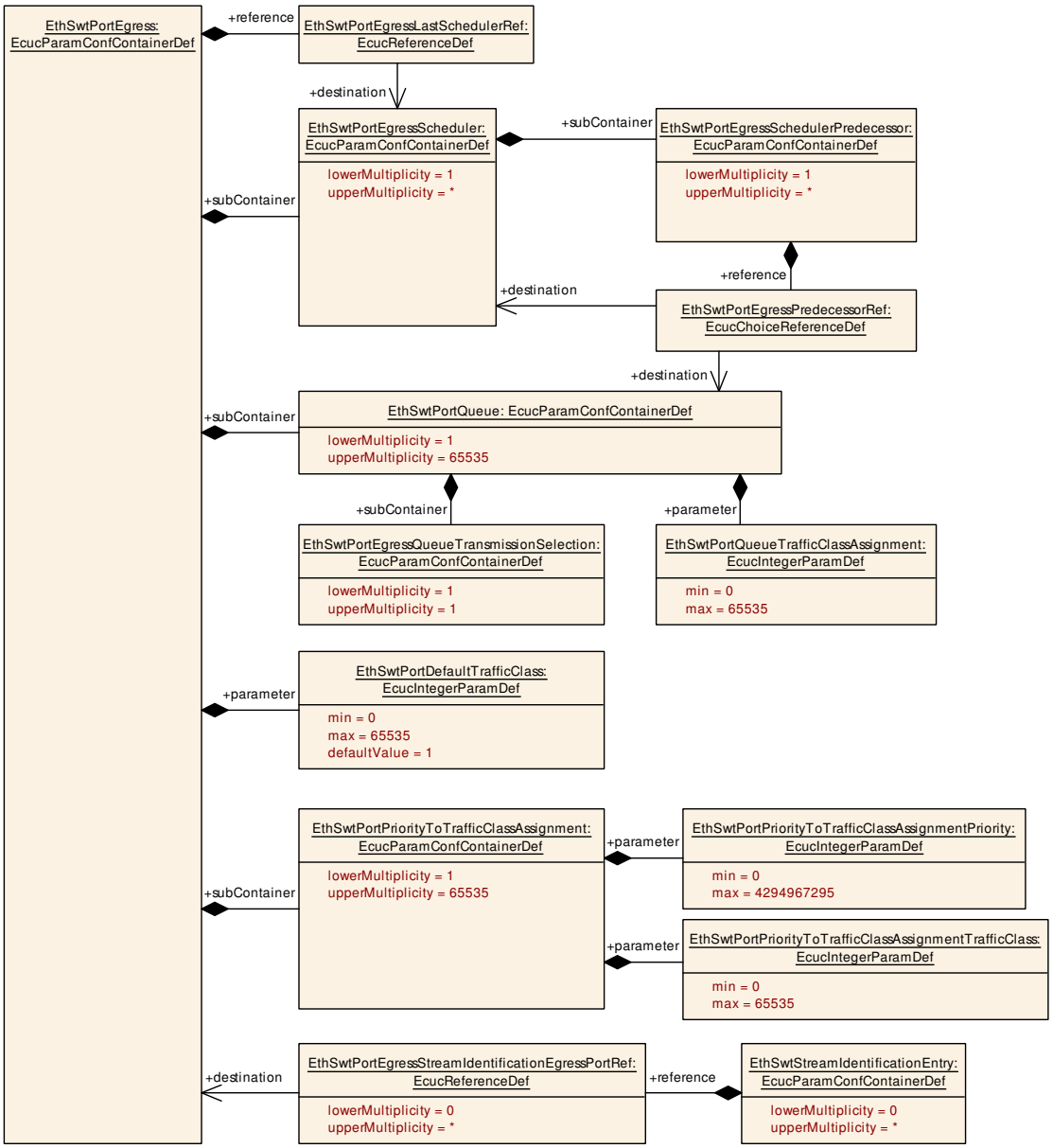

Figure 10.12: `EthSwtPortEgress` (1/3)

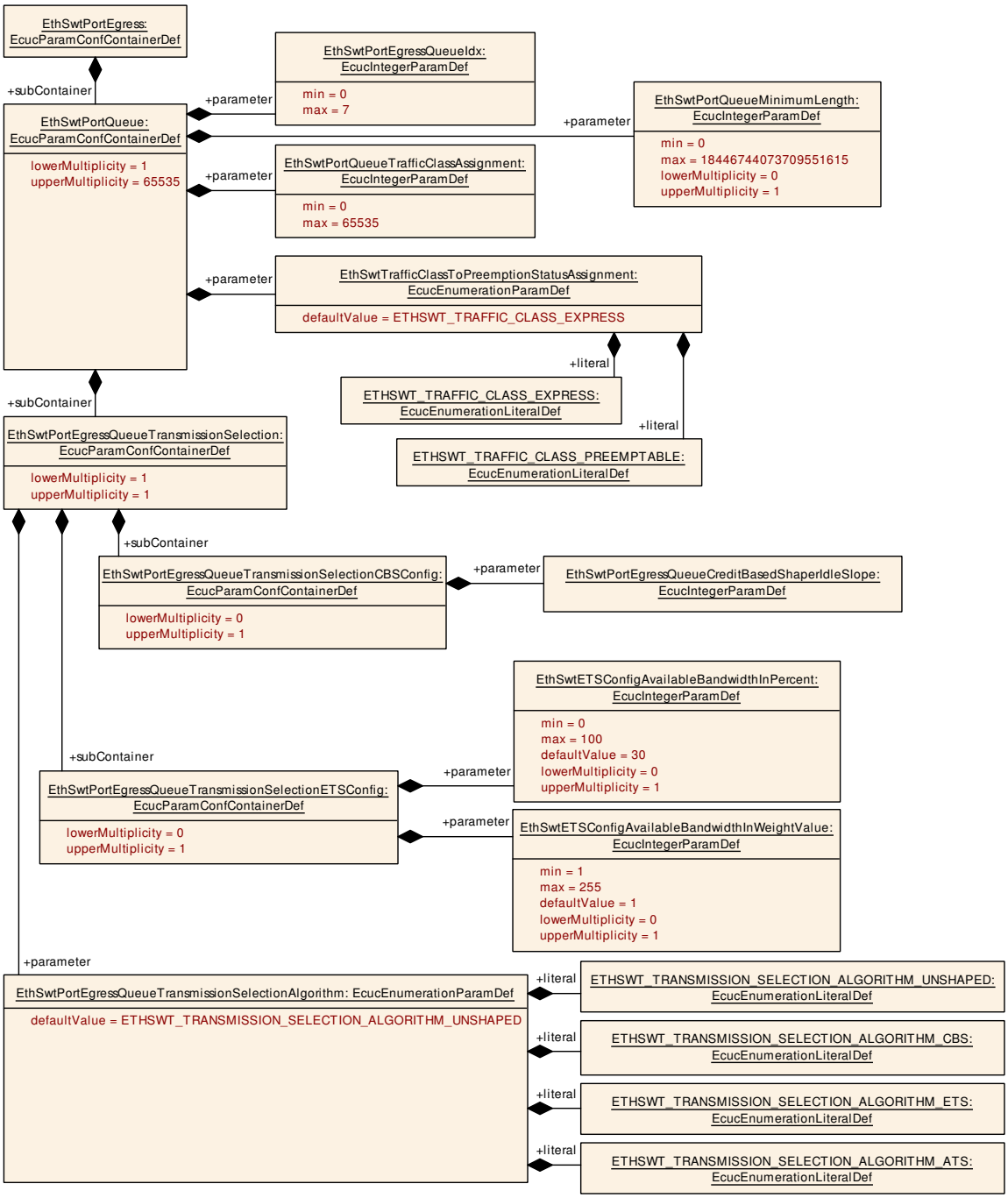

Figure 10.13: `EthSwtPortEgress` (2/3)

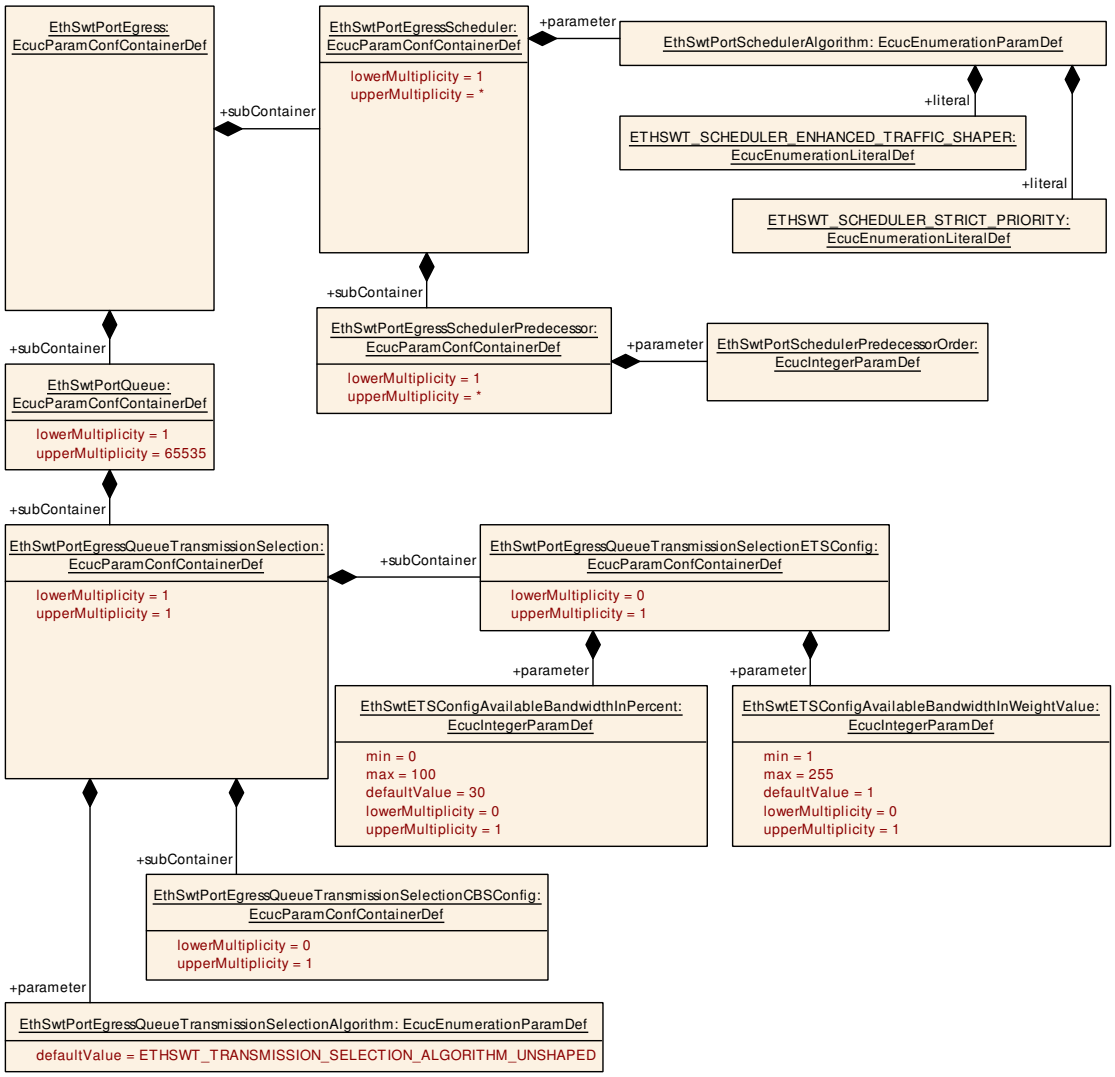

Figure 10.14: `EthSwtPortEgress` (3/3)

10.1.24 EthSwtPortPriorityToTrafficClassAssignment

[ECUC_EthSwt_00248] Definition of EcucParamConfContainerDef EthSwtPortPriorityToTrafficClassAssignment

Status: DRAFT

⌈

Container Name	EthSwtPortPriorityToTrafficClassAssignment
Parent Container	EthSwtPortEgress
Upstream mapping defined	[14, CP_TPS_SystemTemplate]

▽

△

Description	Defines a priority based traffic class assignment. All Ethernet frames with a specific priority (EthSwtPortPriorityToTrafficClassAssignmentPriority) arriving at the egress side within the forwarding process, shall be assigned to the corresponding traffic class (Eth SwtPortPriorityToTrafficClassAssignmentTrafficClass). **Tags:** atp.Status=draft
Multiplicity	1..65535
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
EthSwtPortPriorityToTrafficClassAssignmentPriority	1	[ECUC_EthSwt_00249]
EthSwtPortPriorityToTrafficClassAssignmentTrafficClass	1	[ECUC_EthSwt_00250]

No Included Containers

⌋

[ECUC_EthSwt_00249] Definition of EcucIntegerParamDef EthSwtPortPriorityToTrafficClassAssignmentPriority

Status: DRAFT

⌈

Parameter Name	EthSwtPortPriorityToTrafficClassAssignmentPriority
Parent Container	EthSwtPortPriorityToTrafficClassAssignment
Upstream mapping defined	[14, CP_TPS_SystemTemplate]
Description	Defines the priority derived from the Ethernet frame, which is used to determine the corresponding traffic class, where this Ethernet frame shall be assigned to. The upper value range is restricted to the configured value of EthSwtUsedInternalPriorityUpper Value. **Tags:** atp.Status=draft
Multiplicity	1
Type	EcucIntegerParamDef
Range	0 .. 4294967295
Default value	–
Pre-Compile only	true

⌋

[ECUC_EthSwt_00250] Definition of EcucIntegerParamDef EthSwtPortPriorityToTrafficClassAssignmentTrafficClass

Status: DRAFT

⌈

Parameter Name	EthSwtPortPriorityToTrafficClassAssignmentTrafficClass
Parent Container	EthSwtPortPriorityToTrafficClassAssignment
Upstream mapping defined	[14, CP_TPS_SystemTemplate]

▽

△

Description	Defines the traffic class value where an Ethernet frame, with the corresponding priority, is assigned to. The upper value range is restricted to the configured value of EthSwt UsedTrafficClassUpperValue. **Tags:** atp.Status=draft	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 65535	
Default value	–	
Pre-Compile only	true	

⌋

10.1.25 EthSwtPortEgressScheduler

[ECUC_EthSwt_00017] Definition of EcucParamConfContainerDef EthSwtPortEgressScheduler ⌈

Container Name	EthSwtPortEgressScheduler	
Parent Container	EthSwtPortEgress	
Upstream mapping defined	[14, CP_TPS_SystemTemplate]	
Description	Represents a Scheduler in the egress port.	
Multiplicity	1..*	
Post-Build Multiplicity	Loadable	Selectable
Configuration Parameters		

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
EthSwtPortSchedulerAlgorithm	1	[ECUC_EthSwt_00018]

Included Containers		
Container Name	**Multiplicity**	**Description**
EthSwtPortEgressScheduler Predecessor	1..*	Defines an ordered list of predecessors for this scheduler.

⌋

[ECUC_EthSwt_00018] Definition of EcucEnumerationParamDef EthSwtPortSchedulerAlgorithm ⌈

Parameter Name	EthSwtPortSchedulerAlgorithm
Parent Container	EthSwtPortEgressScheduler
Upstream mapping defined	[14, CP_TPS_SystemTemplate]
Description	Defines the scheduler algorithm.
Multiplicity	1
Type	EcucEnumerationParamDef

▽

△

Range	ETHSWT_SCHEDULER_ ENHANCED_TRAFFIC_SHAPER	Represents a scheduler used for enhanced traffic shaping (e.g. weighted round robin).
	ETHSWT_SCHEDULER_ STRICT_PRIORITY	Represents a strict priority scheduler.
Post-Build Value	Loadable	Selectable

⌋

10.1.26 EthSwtPortEgressSchedulerPredecessor

[ECUC_EthSwt_00019] Definition of EcucParamConfContainerDef EthSwtPortEgressSchedulerPredecessor ⌈

Container Name	EthSwtPortEgressSchedulerPredecessor	
Parent Container	EthSwtPortEgressScheduler	
Upstream mapping defined	[14, CP_TPS_SystemTemplate]	
Description	Defines an ordered list of predecessors for this scheduler.	
Multiplicity	1..*	
Post-Build Multiplicity	Loadable	Selectable
Configuration Parameters		

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
EthSwtPortSchedulerPredecessorOrder	1	[ECUC_EthSwt_00020]
EthSwtPortEgressPredecessorRef	1	[ECUC_EthSwt_00010]

No Included Containers

⌋

[ECUC_EthSwt_00020] Definition of EcucIntegerParamDef EthSwtPortSchedulerPredecessorOrder ⌈

Parameter Name	EthSwtPortSchedulerPredecessorOrder	
Parent Container	EthSwtPortEgressSchedulerPredecessor	
Upstream mapping defined	[14, CP_TPS_SystemTemplate]	
Description	Defines the order of the scheduler predecessors. This value has to be understood as a relative value, i.e. the value shows only the relative ordering of the elements. The highest value has the highest priority and gaps are allowed (not dense based). The values need to be unique within one EthSwtPort Scheduler.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 18446744073709551615	
Default value	–	
Post-Build Value	Loadable	Selectable

⌋

[ECUC_EthSwt_00010] Definition of EcucChoiceReferenceDef EthSwtPortEgressPredecessorRef ⌈

Parameter Name	EthSwtPortEgressPredecessorRef	
Parent Container	EthSwtPortEgressSchedulerPredecessor	
Upstream mapping defined	[14, CP_TPS_SystemTemplate]	
Description	Choice reference to the scheduler predecessor.	
Multiplicity	1	
Type	Choice reference to [EthSwtPortEgressScheduler, EthSwtPortQueue]	
Post-Build Value	Loadable	Selectable

⌋

10.1.27 EthSwtPortQueue

[ECUC_EthSwt_00182] Definition of EcucParamConfContainerDef EthSwtPortQueue

Status: DRAFT

⌈

Container Name	EthSwtPortQueue	
Parent Container	EthSwtPortEgress	
Upstream mapping defined	[14, CP_TPS_SystemTemplate]	
Description	Represents a Queue at the egress port. **Tags:** atp.Status=draft	
Multiplicity	1..65535	
Post-Build Multiplicity	Loadable	Selectable
Configuration Parameters		

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
EthSwtPortEgressQueueIdx	1	[ECUC_EthSwt_00183]
EthSwtPortQueueMinimumLength	0..1	[ECUC_EthSwt_00184]
EthSwtPortQueueTrafficClassAssignment	1	[ECUC_EthSwt_00185]
EthSwtTrafficClassToPreemptionStatusAssignment	1	[ECUC_EthSwt_00255]

Included Containers		
Container Name	**Multiplicity**	**Description**
EthSwtPortEgressQueue TransmissionSelection	1	Represents the transmission selection of an egress port queue. **Tags:** atp.Status=draft

⌋

[ECUC_EthSwt_00183] Definition of EcucIntegerParamDef EthSwtPortEgressQueueIdx

Status: DRAFT

⌈

Parameter Name	EthSwtPortEgressQueueIdx	
Parent Container	EthSwtPortQueue	
Upstream mapping defined	–	
Description	Specifies the instance ID of the queue of the configured Ethernet switch egress port. **Tags:** atp.Status=draft	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 7	
Default value	–	
Post-Build Value	Loadable	Selectable

⌋

[ECUC_EthSwt_00184] Definition of EcucIntegerParamDef EthSwtPortQueueMinimumLength

Status: DRAFT

⌈

Parameter Name	EthSwtPortQueueMinimumLength	
Parent Container	EthSwtPortQueue	
Upstream mapping defined	[14, CP_TPS_SystemTemplate]	
Description	Queue minimum length in Byte. This assignment is used to configure a guaranteed size of a configured Queue. **Tags:** atp.Status=draft	
Multiplicity	0..1	
Type	EcucIntegerParamDef	
Range	0 .. 18446744073709551615	
Default value	–	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

⌋

[ECUC_EthSwt_00185] Definition of EcucIntegerParamDef EthSwtPortQueueTrafficClassAssignment

Status: DRAFT

⌈

Parameter Name	EthSwtPortQueueTrafficClassAssignment	
Parent Container	EthSwtPortQueue	
Upstream mapping defined	[14, CP_TPS_SystemTemplate]	

▽

△

Description	Defines the traffic class where this egress port queue is assigned to. All Ethernet frames which arrive at the corresponding egress port are considered to be enqueued in this egress port queue, where the assigned traffic class of the Ethernet frame match to the assigned traffic class of this egress port queue. **Tags:** atp.Status=draft	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 65535	
Default value	–	
Post-Build Value	Loadable	Selectable

⌋

[ECUC_EthSwt_00255] Definition of EcucEnumerationParamDef EthSwtTrafficClassToPreemptionStatusAssignment

Status: DRAFT

⌈

Parameter Name	EthSwtTrafficClassToPreemptionStatusAssignment	
Parent Container	EthSwtPortQueue	
Upstream mapping defined	[14, CP_TPS_SystemTemplate]	
Description	Defines the preemption status for the traffic class which is derived from the priority via EthSwtPortPriorityToTrafficClassAssignment. If this parameter is set to ETHSWT_TRAFFIC_CLASS_PREEMPTABLE, then the Ethernet frames assigned to the corresponding traffic class could be preempted within the transmission process. If set to ETHSWT_TRAFFIC_CLASS_EXPRESS, then the Ethernet frames assigned to the corresponding traffic class will never be preempted. **Tags:** atp.Status=draft Dependency: If EthSwtFramePreemptionEnable is set to false, then this parameter shall be set to ETHSWT_TRAFFIC_CLASS_EXPRESS.	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	ETHSWT_TRAFFIC_CLASS_EXPRESS	Traffic class will never be preempted. **Tags:** atp.Status=draft
	ETHSWT_TRAFFIC_CLASS_PREEMPTABLE	Traffic class can/may be preempted. **Tags:** atp.Status=draft
Default value	ETHSWT_TRAFFIC_CLASS_EXPRESS	
Post-Build Value	Loadable	Selectable

⌋

10.1.28 EthSwtPortEgressQueueTransmissionSelection

[ECUC_EthSwt_00186] Definition of EcucParamConfContainerDef EthSwtPortEgressQueueTransmissionSelection

Status: DRAFT

⌈

Container Name	EthSwtPortEgressQueueTransmissionSelection
Parent Container	EthSwtPortQueue
Upstream mapping defined	–
Description	Represents the transmission selection of an egress port queue. **Tags:** atp.Status=draft
Multiplicity	1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
EthSwtPortEgressQueueTransmissionSelectionAlgorithm	1	[ECUC_EthSwt_00191]

Included Containers		
Container Name	**Multiplicity**	**Description**
EthSwtPortEgressQueue TransmissionSelectionCBSConfig	0..1	Represents the configuration of a credit based shaper transmission selection algorithm of an egress port queue. This configuration is used if the EthSwtPortEgressQueue TransmissionSelectionAlgorithm is set to ETHSWT_ TRANSMISSION_SELECTION_ALGORITHM_CBS. Lower and upper boundaries of the CBS credit are not configurable by the EthSwt driver. If the credit counter of an implementation should over- or under-run its upper or lower limit at runtime, the resulting behavior is not specified by AUTOSAR or by IEEE and remains implementation specific. **Tags:** atp.Status=draft
EthSwtPortEgressQueue TransmissionSelectionETSConfig	0..1	Represents the configuration of an enhanced transmission selection algorithm of an egress port queue. This configuration is used if the EthSwtPortEgressQueueTransmissionSelection Algorithm is set to ETHSWT_SCHEDULER_ENHANCED_ TRAFFIC_SHAPER. The subordinated configuration parameters allow to configure the ETS in different variants to support commonly used scheduler algorithms (e.g. weighted round robin). **Tags:** atp.Status=draft

⌋

[ECUC_EthSwt_00191] Definition of EcucEnumerationParamDef EthSwtPort EgressQueueTransmissionSelectionAlgorithm

Status: DRAFT

⌈

Parameter Name	EthSwtPortEgressQueueTransmissionSelectionAlgorithm
Parent Container	EthSwtPortEgressQueueTransmissionSelection
Upstream mapping defined	–
Description	Represents the transmission selection algorithm of an egress port queue. **Tags:** atp.Status=draft
Multiplicity	1
Type	EcucEnumerationParamDef

▽

△

Range	ETHSWT_TRANSMISSION_SELECTION_ALGORITHM_ATS	Ethernet frames are selected from the egress queue for transmission according the asynchronous traffic shaping algorithm. **Tags:** atp.Status=draft
	ETHSWT_TRANSMISSION_SELECTION_ALGORITHM_CBS	Ethernet frames are selected from the egress queue for transmission according the credit based shaping algorithm. **Tags:** atp.Status=draft
	ETHSWT_TRANSMISSION_SELECTION_ALGORITHM_ETS	Ethernet frames are selected from the egress queue for transmission according the enhanced transmission selection algorithm. **Tags:** atp.Status=draft
	ETHSWT_TRANSMISSION_SELECTION_ALGORITHM_UNSHAPED	Ethernet frames are selected from the egress queue for transmission in an unshaped manner. Please note: IEEE802.1Q uses the term "strict priority". Term "UNSHAPED" is used to avoid confusion with strict priority in context of EthSwt PortEgressScheduler. **Tags:** atp.Status=draft
Default value	ETHSWT_TRANSMISSION_SELECTION_ALGORITHM_UNSHAPED	
Post-Build Value	Loadable	Selectable

⌋

10.1.29 EthSwtPortEgressQueueTransmissionSelectionCBSConfig

[ECUC_EthSwt_00187] Definition of EcucParamConfContainerDef EthSwtPortEgressQueueTransmissionSelectionCBSConfig

Status: DRAFT

⌈

Container Name	EthSwtPortEgressQueueTransmissionSelectionCBSConfig	
Parent Container	EthSwtPortEgressQueueTransmissionSelection	
Upstream mapping defined	[14, CP_TPS_SystemTemplate]	
Description	Represents the configuration of a credit based shaper transmission selection algorithm of an egress port queue. This configuration is used if the EthSwtPortEgressQueueTransmissionSelection Algorithm is set to ETHSWT_TRANSMISSION_SELECTION_ALGORITHM_CBS. Lower and upper boundaries of the CBS credit are not configurable by the EthSwt driver. If the credit counter of an implementation should over- or under-run its upper or lower limit at runtime, the resulting behavior is not specified by AUTOSAR or by IEEE and remains implementation specific. **Tags:** atp.Status=draft	
Multiplicity	0..1	
Post-Build Multiplicity	Loadable	Selectable
Configuration Parameters		

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
EthSwtPortEgressQueueCreditBasedShaperIdleSlope	1	[ECUC_EthSwt_00188]

No Included Containers

⌋

[ECUC_EthSwt_00188] Definition of EcucIntegerParamDef EthSwtPortEgressQueueCreditBasedShaperIdleSlope

Status: DRAFT

⌈

Parameter Name	EthSwtPortEgressQueueCreditBasedShaperIdleSlope	
Parent Container	EthSwtPortEgressQueueTransmissionSelectionCBSConfig	
Upstream mapping defined	–	
Description	Defines the increase of credit in bits per second for the AVB shaper. Note: this parameter maps to IEEE802.1Q parameter "ieee8021FqtssAdminIdleSlope Ms" and "ieee8021FqtssAdminIdleSlopeLs". Note: If the value exceeds the available bandwidth according the configured EthSwt PortPhysicalLayerType, the CBS becomes ineffective. **Tags:** atp.Status=draft	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 18446744073709551615	
Default value	–	
Post-Build Value	Loadable	Selectable

⌋

10.1.30 EthSwtPortEgressQueueTransmissionSelectionETSConfig

[ECUC_EthSwt_00251] Definition of EcucParamConfContainerDef EthSwtPortEgressQueueTransmissionSelectionETSConfig

Status: DRAFT

⌈

Container Name	EthSwtPortEgressQueueTransmissionSelectionETSConfig
Parent Container	EthSwtPortEgressQueueTransmissionSelection
Upstream mapping defined	[14, CP_TPS_SystemTemplate]
Description	Represents the configuration of an enhanced transmission selection algorithm of an egress port queue. This configuration is used if the EthSwtPortEgressQueue TransmissionSelectionAlgorithm is set to ETHSWT_SCHEDULER_ENHANCED_ TRAFFIC_SHAPER. The subordinated configuration parameters allow to configure the ETS in different variants to support commonly used scheduler algorithms (e.g. weighted round robin). **Tags:** atp.Status=draft
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
EthSwtETSConfigAvailableBandwidthInPercent	0..1	[ECUC_EthSwt_00252]
EthSwtETSConfigAvailableBandwidthInWeightValue	0..1	[ECUC_EthSwt_00253]

No Included Containers

⌋

[ECUC_EthSwt_00252] Definition of EcucIntegerParamDef EthSwtETSConfig AvailableBandwidthInPercent

Status: DRAFT

⌈

Parameter Name	EthSwtETSConfigAvailableBandwidthInPercent	
Parent Container	EthSwtPortEgressQueueTransmissionSelectionETSConfig	
Upstream mapping defined	[14, CP_TPS_SystemTemplate]	
Description	Represents the configuration of an enhanced transmission selection algorithm for one egress port queue, where the available bandwidth is configured in percent. The percent value represents the available bandwidth for emission opportunities to transmit Ethernet frames calculated in bits. The resolution is 1%. **Tags:** atp.Status=draft	
Multiplicity	0..1	
Type	EcucIntegerParamDef	
Range	0 .. 100	
Default value	30	
Pre-Compile only	true	

⌋

[ECUC_EthSwt_00253] Definition of EcucIntegerParamDef EthSwtETSConfig AvailableBandwidthInWeightValue

Status: DRAFT

⌈

Parameter Name	EthSwtETSConfigAvailableBandwidthInWeightValue	
Parent Container	EthSwtPortEgressQueueTransmissionSelectionETSConfig	
Upstream mapping defined	[14, CP_TPS_SystemTemplate]	
Description	Represents the configuration of an enhanced transmission selection algorithm of an egress port queue, where the available bandwidth is configured as weight value. The weight value represents the number of emission opportunities to transmit Ethernet frames. **Tags:** atp.Status=draft	
Multiplicity	0..1	
Type	EcucIntegerParamDef	
Range	1 .. 255	
Default value	1	

▽

△

Pre-Compile only	true

⌋

10.1.31 EthSwtPortIngress

[ECUC_EthSwt_00014] Definition of EcucParamConfContainerDef EthSwtPortIngress ⌈

Container Name	EthSwtPortIngress
Parent Container	EthSwtPort
Upstream mapping defined	–
Description	Configuration of one Ethernet Switch Port ingress behavior.
Multiplicity	1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
EthSwtPortIngressDefaultPriority	0..1	[ECUC_EthSwt_00096]
EthSwtPortIngressDefaultVlan	0..1	[ECUC_EthSwt_00095]
EthSwtPortIngressDropUntagged	1	[ECUC_EthSwt_00097]

Included Containers		
Container Name	**Multiplicity**	**Description**
EthSwtPortIngressScheduler	0..*	Represents a Scheduler configuration at an ingress port. **Tags:** atp.Status=draft
EthSwtPortIngressVlanTranslationTable	0..1	This container defines 0..* entries of the form (IngressVlanID, TranslatedVlanID) that define the ingress Vlan translation. The IngressVlanID is the VlanID read from the incoming frame upon reception (ingress), which is replaced by the corresponding TranslatedVlanID upon ingress Vlan translation.
EthSwtPortPriorityRegeneration	8	Defines a priority regeneration where the EthSwtPortPriority RegenerationIngressPCP is replaced by EthSwtPortPriority RegenerationRegeneratedPriority. The EthSwtPortPriorityRegeneration is mandatory and shall always be available. An EthSwtPortPriorityRegeneration shall have 8 mappings, one for each priority. Rational: an Ethernet switch always performs a priority regeneration.

⌋

[ECUC_EthSwt_00096] Definition of EcucIntegerParamDef EthSwtPortIngress DefaultPriority ⌈

Parameter Name	EthSwtPortIngressDefaultPriority	
Parent Container	EthSwtPortIngress	
Upstream mapping defined	[14, CP_TPS_SystemTemplate]	
Description	Default priority for ingress. Dependency: If EthSwtPortIngressDefaultPriority is configured (multiplicity set to 1) then EthSwtPortIngressDefaultVlan shall be configured. If EthSwtPortIngressDefault Vlan is configured EthSwtPortIngressDropUntagged shall be set to FALSE.	
Multiplicity	0..1	
Type	EcucIntegerParamDef	
Range	0 .. 7	
Default value	0	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

⌋

[ECUC_EthSwt_00095] Definition of EcucIntegerParamDef EthSwtPortIngress DefaultVlan ⌈

Parameter Name	EthSwtPortIngressDefaultVlan	
Parent Container	EthSwtPortIngress	
Upstream mapping defined	[14, CP_TPS_SystemTemplate]	
Description	Default VLAN for ingress. Dependency: If EthSwtPortIngressDefaultVlan is configured (multiplicity set to 1) then EthSwtPortIngressDefaultPriority shall be configured. If EthSwtPortIngressDefaultVlan is configured EthSwtPortIngressDropUntagged shall be set to FALSE.	
Multiplicity	0..1	
Type	EcucIntegerParamDef	
Range	0 .. 4094	
Default value	1	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

⌋

[ECUC_EthSwt_00097] Definition of EcucBooleanParamDef EthSwtPortIngress DropUntagged ⌈

Parameter Name	EthSwtPortIngressDropUntagged
Parent Container	EthSwtPortIngress
Upstream mapping defined	[14, CP_TPS_SystemTemplate]
Description	Defines the ingress behavior for untagged frames. Dependency: If EthSwtPortIngressDropUntagged is set to TRUE then EthSwtPort IngressDefaultVlan and EthSwtPortIngressDefaultPriority parameters shall not be configured.
Multiplicity	1
Type	EcucBooleanParamDef

▽

△

Default value	false	
Post-Build Value	Loadable	Selectable

┘

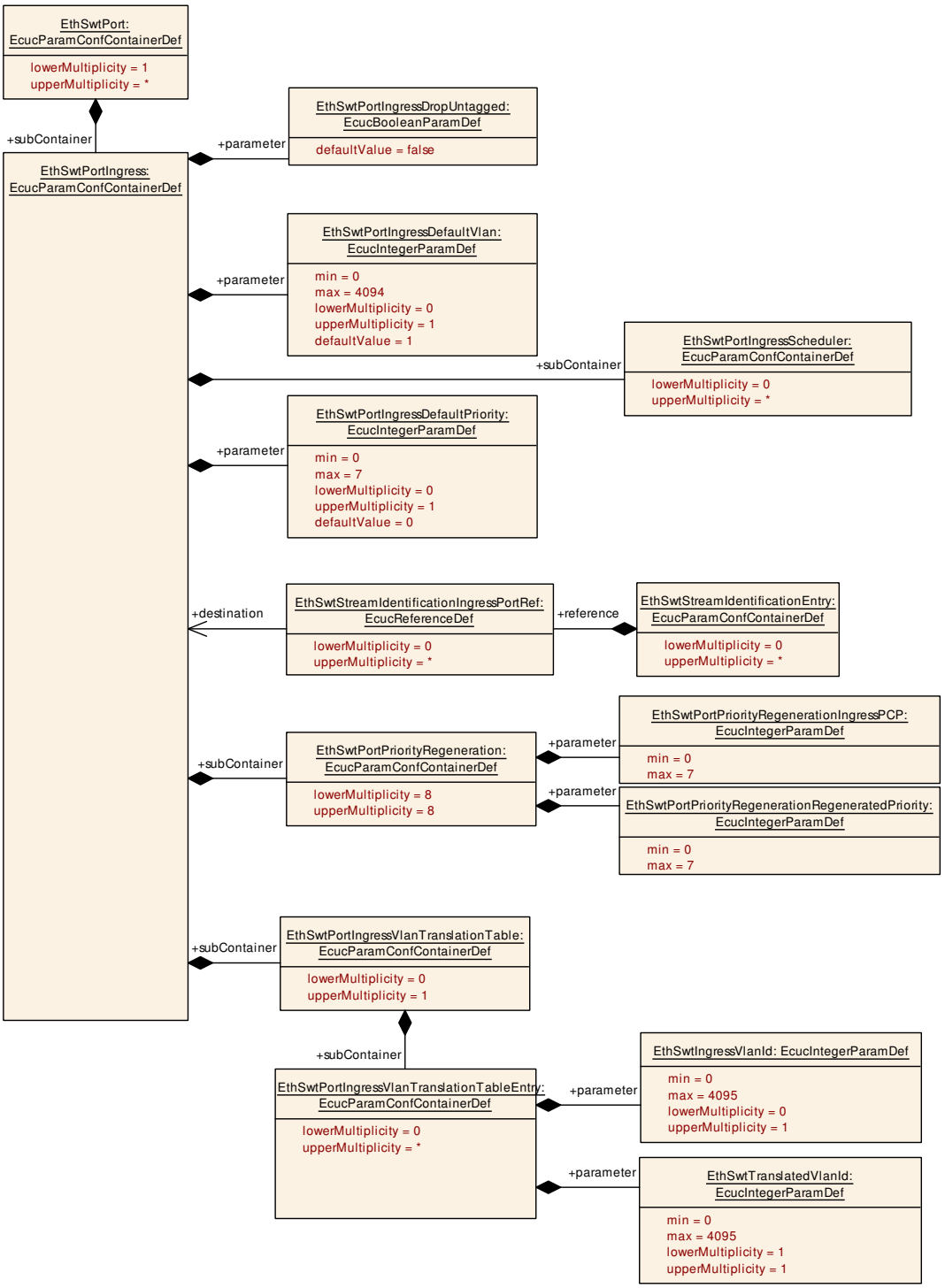

Figure 10.15: `EthSwtPortIngress` (1/2)

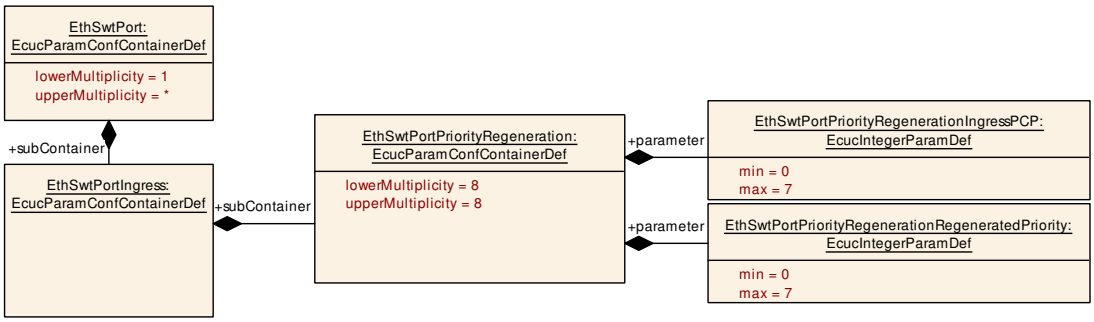

Figure 10.16: `EthSwtPortIngress` (2/2)

10.1.32 EthSwtPortIngressScheduler

[ECUC_EthSwt_00139] Definition of EcucParamConfContainerDef EthSwtPortIngressScheduler

Status: DRAFT

⌈

Container Name	EthSwtPortIngressScheduler	
Parent Container	EthSwtPortIngress	
Upstream mapping defined	–	
Description	Represents a Scheduler configuration at an ingress port. **Tags:** atp.Status=draft	
Multiplicity	0..*	
Post-Build Multiplicity	Loadable	Selectable
Configuration Parameters		

No Included Parameters

No Included Containers

⌋

10.1.33 EthSwtPortIngressVlanTranslationTable

[ECUC_EthSwt_00256] Definition of EcucParamConfContainerDef EthSwtPortIngressVlanTranslationTable ⌈

Container Name	EthSwtPortIngressVlanTranslationTable
Parent Container	EthSwtPortIngress
Upstream mapping defined	[14, CP_TPS_SystemTemplate]

▽

△

Description	This container defines 0..* entries of the form (IngressVlanID, TranslatedVlanID) that define the ingress Vlan translation. The IngressVlanID is the VlanID read from the incoming frame upon reception (ingress), which is replaced by the corresponding TranslatedVlanID upon ingress Vlan translation.	
Multiplicity	0..1	
Post-Build Multiplicity	-	Selectable
Configuration Parameters		

No Included Parameters

Included Containers		
Container Name	**Multiplicity**	**Description**
EthSwtPortIngressVlanTranslationTableEntry	0..*	This container defines the mapping between the IngressVlanID (VlanID read from the received frame upon ingress) and the corresponding TranslatedVlanID upon ingress Vlan translation.

⌋

10.1.34 EthSwtPortIngressVlanTranslationTableEntry

[ECUC_EthSwt_00257] Definition of EcucParamConfContainerDef EthSwtPortIngressVlanTranslationTableEntry ⌈

Container Name	EthSwtPortIngressVlanTranslationTableEntry	
Parent Container	EthSwtPortIngressVlanTranslationTable	
Upstream mapping defined	[14, CP_TPS_SystemTemplate]	
Description	This container defines the mapping between the IngressVlanID (VlanID read from the received frame upon ingress) and the corresponding TranslatedVlanID upon ingress Vlan translation.	
Multiplicity	0..*	
Post-Build Multiplicity	Loadable	Selectable
Configuration Parameters		

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
EthSwtIngressVlanId	0..1	[ECUC_EthSwt_00258]
EthSwtTranslatedVlanId	1	[ECUC_EthSwt_00259]

No Included Containers

⌋

[ECUC_EthSwt_00258] Definition of EcucIntegerParamDef EthSwtIngressVlanId ⌈

Parameter Name	EthSwtIngressVlanId	
Parent Container	EthSwtPortIngressVlanTranslationTableEntry	
Upstream mapping defined	[14, CP_TPS_SystemTemplate]	
Description	Incoming VlanID from received frame.	
Multiplicity	0..1	
Type	EcucIntegerParamDef	
Range	0 .. 4095	
Default value	–	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

⌋

[ECUC_EthSwt_00259] Definition of EcucIntegerParamDef EthSwtTranslatedVlanId ⌈

Parameter Name	EthSwtTranslatedVlanId	
Parent Container	EthSwtPortIngressVlanTranslationTableEntry	
Upstream mapping defined	[14, CP_TPS_SystemTemplate]	
Description	Mapped VlanID after ingress Vlan translation.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 4095	
Default value	–	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

⌋

10.1.35 EthSwtPortPriorityRegeneration

[ECUC_EthSwt_00057] Definition of EcucParamConfContainerDef EthSwtPortPriorityRegeneration ⌈

Container Name	EthSwtPortPriorityRegeneration
Parent Container	EthSwtPortIngress
Upstream mapping defined	[14, CP_TPS_SystemTemplate]
Description	Defines a priority regeneration where the EthSwtPortPriorityRegenerationIngressPCP is replaced by EthSwtPortPriorityRegenerationRegeneratedPriority. The EthSwtPortPriorityRegeneration is mandatory and shall always be available. An EthSwtPortPriorityRegeneration shall have 8 mappings, one for each priority. Rational: an Ethernet switch always performs a priority regeneration.
Multiplicity	8
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
EthSwtPortPriorityRegenerationIngressPCP	1	[ECUC_EthSwt_00058]
EthSwtPortPriorityRegenerationRegeneratedPriority	1	[ECUC_EthSwt_00059]

No Included Containers

⌋

[ECUC_EthSwt_00058] Definition of EcucIntegerParamDef EthSwtPortPriorityRegenerationIngressPCP ⌈

Parameter Name	EthSwtPortPriorityRegenerationIngressPCP	
Parent Container	EthSwtPortPriorityRegeneration	
Upstream mapping defined	[14, CP_TPS_SystemTemplate]	
Description	PCP (VLAN-priority) in the incoming message.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 7	
Default value	–	
Post-Build Value	Loadable	Selectable

⌋

[ECUC_EthSwt_00059] Definition of EcucIntegerParamDef EthSwtPortPriorityRegenerationRegeneratedPriority ⌈

Parameter Name	EthSwtPortPriorityRegenerationRegeneratedPriority	
Parent Container	EthSwtPortPriorityRegeneration	
Upstream mapping defined	[14, CP_TPS_SystemTemplate]	
Description	Message priority the incoming message will be tagged with.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 7	
Default value	–	
Post-Build Value	Loadable	Selectable

⌋

10.1.36 EthSwtSpi

[ECUC_EthSwt_00030] Definition of EcucParamConfContainerDef EthSwtSpi ⌈

Container Name	EthSwtSpi
Parent Container	EthSwtConfig
Upstream mapping defined	–
Description	Configuration of one Ethernet Switch SPI access (if SPI is used).

▽

△

Multiplicity	0..1	
Post-Build Multiplicity	-	Selectable
Configuration Parameters		

No Included Parameters

Included Containers		
Container Name	**Multiplicity**	**Description**
EthSwtSpiSequence	1..*	Container gives EthSwt driver information about one SPI sequence. One SPI sequence used by EthSwt driver is in exclusive use for it. No other driver is allowed to access this sequence. EthSwt driver may use one sequence to access n Eth Swt hardware chips of the same type or n sequences are used to access one single EthSwt hardware chip. If a EthSwt hardware has no SPI interface, there is no instance of this container.

⌋

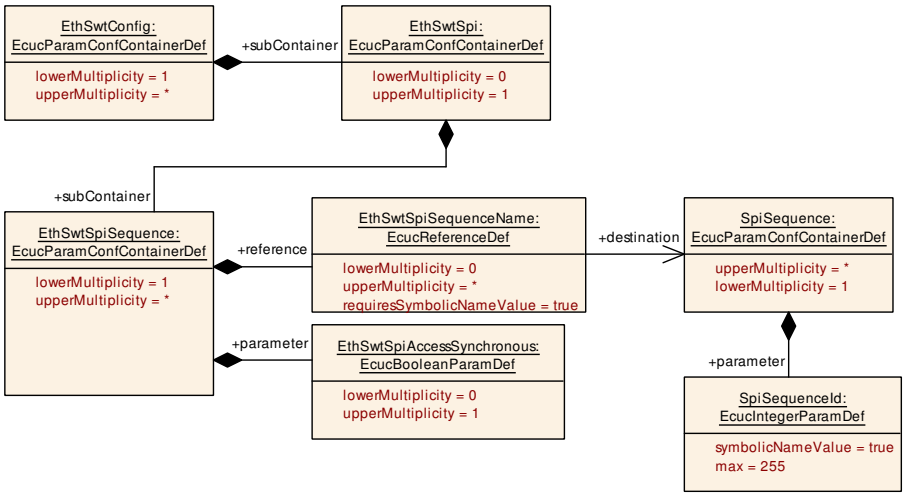

Figure 10.17: `EthSwtSpi`

10.1.37 EthSwtSpiSequence

[ECUC_EthSwt_00034] Definition of EcucParamConfContainerDef EthSwtSpiSequence ⌈

Container Name	EthSwtSpiSequence
Parent Container	EthSwtSpi
Upstream mapping defined	–
Description	Container gives EthSwt driver information about one SPI sequence. One SPI sequence used by EthSwt driver is in exclusive use for it. No other driver is allowed to access this sequence. EthSwt driver may use one sequence to access n EthSwt hardware chips of the same type or n sequences are used to access one single EthSwt hardware chip. If a EthSwt hardware has no SPI interface, there is no instance of this container.

▽

△

Multiplicity	1..*	
Post-Build Multiplicity	-	Selectable
Configuration Parameters		

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
EthSwtSpiAccessSynchronous	0..1	[ECUC_EthSwt_00036]
EthSwtSpiSequenceName	0..*	[ECUC_EthSwt_00035]

No Included Containers

⌋

[ECUC_EthSwt_00036] Definition of EcucBooleanParamDef EthSwtSpiAccessSynchronous ⌈

Parameter Name	EthSwtSpiAccessSynchronous	
Parent Container	EthSwtSpiSequence	
Upstream mapping defined	–	
Description	This parameter is used to define whether the access to the Spi sequence is synchronous or asynchronous. true: SPI access is synchronous. false: SPI access is asynchronous.	
Multiplicity	0..1	
Type	EcucBooleanParamDef	
Default value	–	
Post-Build Multiplicity	-	Selectable
Post-Build Value	-	Selectable

⌋

[ECUC_EthSwt_00035] Definition of EcucReferenceDef EthSwtSpiSequenceName ⌈

Parameter Name	EthSwtSpiSequenceName	
Parent Container	EthSwtSpiSequence	
Upstream mapping defined	–	
Description	Reference to a Spi sequence configuration container.	
Multiplicity	0..*	
Type	Symbolic name reference to SpiSequence	
Post-Build Multiplicity	-	Selectable
Post-Build Value	-	Selectable

⌋

10.1.38 EthSwtStreamIdentificationTable

[ECUC_EthSwt_00208] Definition of EcucParamConfContainerDef EthSwtStream IdentificationTable

Status: DRAFT

⌈

Container Name	EthSwtStreamIdentificationTable	
Parent Container	EthSwtConfig	
Upstream mapping defined	–	
Description	Configuration of a stream identification table. **Tags:** atp.Status=draft	
Multiplicity	0..1	
Post-Build Multiplicity	Loadable	Selectable
Configuration Parameters		

No Included Parameters

Included Containers		
Container Name	**Multiplicity**	**Description**
EthSwtStreamIdentificationEntry	0..*	Configuration of a stream identification. **Tags:** atp.Status=draft

⌋

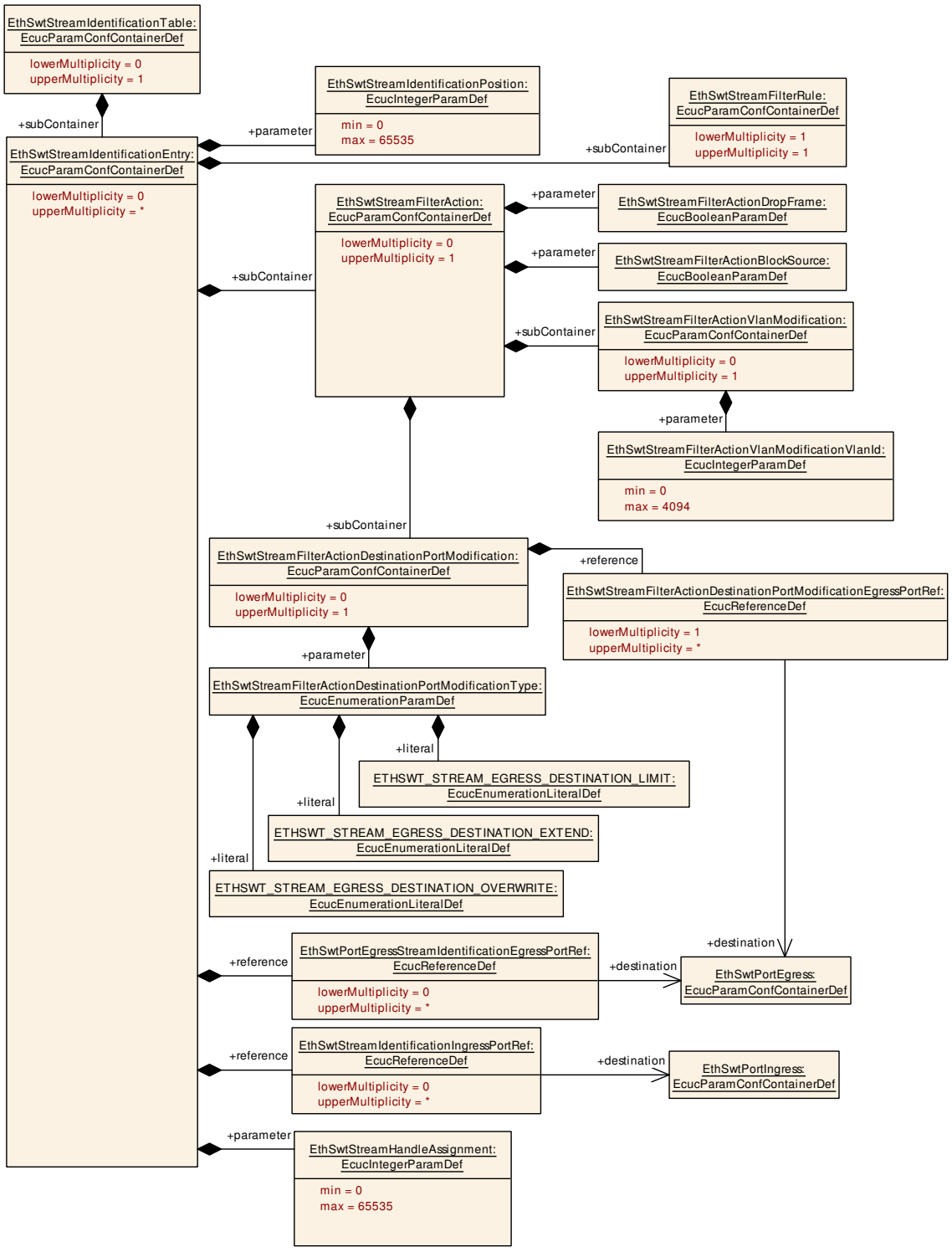

Figure 10.18: EthSwtStreamIdentificationTable

10.1.39 EthSwtStreamIdentificationEntry

[ECUC_EthSwt_00140] Definition of EcucParamConfContainerDef EthSwtStream IdentificationEntry

Status: DRAFT

⌈

Container Name	EthSwtStreamIdentificationEntry	
Parent Container	EthSwtStreamIdentificationTable	
Upstream mapping defined	[14, CP_TPS_SystemTemplate]	
Description	Configuration of a stream identification. **Tags:** atp.Status=draft	
Multiplicity	0..*	
Post-Build Multiplicity	Loadable	Selectable
Configuration Parameters		

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
EthSwtStreamHandleAssignment	1	[ECUC_EthSwt_00211]
EthSwtStreamIdentificationPosition	1	[ECUC_EthSwt_00142]
EthSwtPortEgressStreamIdentificationEgressPortRef	0..*	[ECUC_EthSwt_00153]
EthSwtStreamIdentificationIngressPortRef	0..*	[ECUC_EthSwt_00152]

Included Containers		
Container Name	**Multiplicity**	**Description**
EthSwtStreamFilterAction	0..1	Configuration of a stream filter action. **Tags:** atp.Status=draft
EthSwtStreamFilterRule	1	Configuration of a filter rule. **Tags:** atp.Status=draft

⌋

[ECUC_EthSwt_00211] Definition of EcucIntegerParamDef EthSwtStreamHandle Assignment

Status: DRAFT

⌈

Parameter Name	EthSwtStreamHandleAssignment	
Parent Container	EthSwtStreamIdentificationEntry	
Upstream mapping defined	–	
Description	Assigment of this stream identification to an stream filter entry. **Tags:** atp.Status=draft	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 65535	
Default value	–	

▽

△

Post-Build Value	Loadable	Selectable

⌋

[ECUC_EthSwt_00142] Definition of EcucIntegerParamDef EthSwtStreamIdentificationPosition

Status: DRAFT

⌈

Parameter Name	EthSwtStreamIdentificationPosition	
Parent Container	EthSwtStreamIdentificationEntry	
Upstream mapping defined	–	
Description	Specifies the position as unique ID within an ordered list of EthSwtStreamIdentification Entrys. The ordered list shall start with 0 and continue as linear list with no gaps. Note: The list is processed in ascending order. The instance of EthSwtStream IdentificationEntry with position 0 is processed first. **Tags:** atp.Status=draft	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 65535	
Default value	–	
Post-Build Value	Loadable	Selectable

⌋

[ECUC_EthSwt_00153] Definition of EcucReferenceDef EthSwtPortEgressStreamIdentificationEgressPortRef

Status: DRAFT

⌈

Parameter Name	EthSwtPortEgressStreamIdentificationEgressPortRef	
Parent Container	EthSwtStreamIdentificationEntry	
Upstream mapping defined	–	
Description	Reference to the egress ports this stream identification applies to. **Tags:** atp.Status=draft	
Multiplicity	0..*	
Type	Reference to EthSwtPortEgress	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

⌋

[ECUC_EthSwt_00152] Definition of EcucReferenceDef EthSwtStreamIdentificationIngressPortRef

Status: DRAFT

⌈

Parameter Name	EthSwtStreamIdentificationIngressPortRef	
Parent Container	EthSwtStreamIdentificationEntry	
Upstream mapping defined	–	
Description	Reference to the ingress ports this stream identification applies to. **Tags:** atp.Status=draft	
Multiplicity	0..*	
Type	Reference to EthSwtPortIngress	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

⌋

10.1.40 EthSwtStreamFilterAction

[ECUC_EthSwt_00143] Definition of EcucParamConfContainerDef EthSwtStreamFilterAction

Status: DRAFT

⌈

Container Name	EthSwtStreamFilterAction	
Parent Container	EthSwtStreamIdentificationEntry	
Upstream mapping defined	[14, CP_TPS_SystemTemplate]	
Description	Configuration of a stream filter action. **Tags:** atp.Status=draft	
Multiplicity	0..1	
Post-Build Multiplicity	Loadable	Selectable
Configuration Parameters		

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
EthSwtStreamFilterActionBlockSource	1	[ECUC_EthSwt_00145]
EthSwtStreamFilterActionDropFrame	1	[ECUC_EthSwt_00144]

Included Containers		
Container Name	**Multiplicity**	**Description**
EthSwtStreamFilterAction DestinationPortModification	0..1	Defines the action to modify the destination port vector determined by the frame forwarding process for a particular Ethernet frame. The egress destination port vector of an Ethernet frame can be extended, limited, or overwritten. **Tags:** atp.Status=draft

▽

△

Included Containers		
Container Name	**Multiplicity**	**Description**
EthSwtStreamFilterActionVlan Modification	0..1	Defines the action to modify the VLAN-ID within a VLAN-tag of an Ethernet frame. **Tags:** atp.Status=draft

⌋

[ECUC_EthSwt_00145] Definition of EcucBooleanParamDef EthSwtStreamFilter ActionBlockSource

Status: DRAFT

⌈

Parameter Name	EthSwtStreamFilterActionBlockSource	
Parent Container	EthSwtStreamFilterAction	
Upstream mapping defined	[14, CP_TPS_SystemTemplate]	
Description	Enables Blocking all frames from the MAC address. **Tags:** atp.Status=draft	
Multiplicity	1	
Type	EcucBooleanParamDef	
Default value	–	
Post-Build Value	Loadable	Selectable

⌋

[ECUC_EthSwt_00144] Definition of EcucBooleanParamDef EthSwtStreamFilter ActionDropFrame

Status: DRAFT

⌈

Parameter Name	EthSwtStreamFilterActionDropFrame	
Parent Container	EthSwtStreamFilterAction	
Upstream mapping defined	[14, CP_TPS_SystemTemplate]	
Description	Enables Drop Frame action. **Tags:** atp.Status=draft	
Multiplicity	1	
Type	EcucBooleanParamDef	
Default value	–	
Post-Build Value	Loadable	Selectable

⌋

10.1.41 EthSwtStreamFilterActionDestinationPortModification

[ECUC_EthSwt_00148] Definition of EcucParamConfContainerDef EthSwtStream FilterActionDestinationPortModification

Status: DRAFT

⌈

Container Name	EthSwtStreamFilterActionDestinationPortModification	
Parent Container	EthSwtStreamFilterAction	
Upstream mapping defined	[14, CP_TPS_SystemTemplate]	
Description	Defines the action to modify the destination port vector determined by the frame forwarding process for a particular Ethernet frame. The egress destination port vector of an Ethernet frame can be extended, limited, or overwritten. **Tags:** atp.Status=draft	
Multiplicity	0..1	
Post-Build Multiplicity	Loadable	Selectable
Configuration Parameters		

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
EthSwtStreamFilterActionDestinationPortModificationType	1	[ECUC_EthSwt_00150]
EthSwtStreamFilterActionDestinationPortModification EgressPortRef	1..*	[ECUC_EthSwt_00149]

No Included Containers

⌋

[ECUC_EthSwt_00150] Definition of EcucEnumerationParamDef EthSwtStream FilterActionDestinationPortModificationType

Status: DRAFT

⌈

Parameter Name	EthSwtStreamFilterActionDestinationPortModificationType	
Parent Container	EthSwtStreamFilterActionDestinationPortModification	
Upstream mapping defined	[14, CP_TPS_SystemTemplate]	
Description	Defines the method to modify the egress destination port vector. **Tags:** atp.Status=draft	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	ETHSWT_STREAM_EGRESS_DESTINATION_EXTEND	Extend the egress destination port vector of an Ethernet frame with the configured egress destination port(s). This corresponds to the union of the existing egress destination port vector and the configured egress destination port(s) **Tags:** atp.Status=draft

▽

△

	ETHSWT_STREAM_EGRESS_DESTINATION_LIMIT	Limit the egress destination port vector of an Ethernet frame to the configured egress destination port(s). This corresponds to the intersection of the existing egress destination port vector and the configured egress destination port(s) **Tags:** atp.Status=draft
	ETHSWT_STREAM_EGRESS_DESTINATION_OVERWRITE	Overwrite the egress destination port vector of an Ethernet frame with the configured egress destination port(s). This results in the existing egress destination port vector being replaced by the configured egress destination port(s). **Tags:** atp.Status=draft
Post-Build Value	Loadable	Selectable

⌋

[ECUC_EthSwt_00149] Definition of EcucReferenceDef EthSwtStreamFilterActionDestinationPortModificationEgressPortRef

Status: DRAFT

⌈

Parameter Name	EthSwtStreamFilterActionDestinationPortModificationEgressPortRef	
Parent Container	EthSwtStreamFilterActionDestinationPortModification	
Upstream mapping defined	[14, CP_TPS_SystemTemplate]	
Description	Defines a set of destination ports (egress ports) used for the modification of the egress destination of an Ethernet frame. **Tags:** atp.Status=draft	
Multiplicity	1..*	
Type	Reference to EthSwtPortEgress	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

⌋

10.1.42 EthSwtStreamFilterActionVlanModification

[ECUC_EthSwt_00146] Definition of EcucParamConfContainerDef EthSwtStreamFilterActionVlanModification

Status: DRAFT

⌈

Container Name	EthSwtStreamFilterActionVlanModification
Parent Container	EthSwtStreamFilterAction
Upstream mapping defined	[14, CP_TPS_SystemTemplate]

▽

△

Description	Defines the action to modify the VLAN-ID within a VLAN-tag of an Ethernet frame. **Tags:** atp.Status=draft	
Multiplicity	0..1	
Post-Build Multiplicity	Loadable	Selectable
Configuration Parameters		

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
EthSwtStreamFilterActionVlanModificationVlanId	1	[ECUC_EthSwt_00147]

No Included Containers

⌋

[ECUC_EthSwt_00147] Definition of EcucIntegerParamDef EthSwtStreamFilterActionVlanModificationVlanId

Status: DRAFT

⌈

Parameter Name	EthSwtStreamFilterActionVlanModificationVlanId	
Parent Container	EthSwtStreamFilterActionVlanModification	
Upstream mapping defined	–	
Description	Defines the VLAN-ID to modify the existing VLAN-ID within the VLAN-tag of an Ethernet frame. **Tags:** atp.Status=draft	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 4094	
Default value	–	
Post-Build Value	Loadable	Selectable

⌋

10.1.43 EthSwtStreamFilterRule

[ECUC_EthSwt_00141] Definition of EcucParamConfContainerDef EthSwtStreamFilterRule

Status: DRAFT

⌈

Container Name	EthSwtStreamFilterRule
Parent Container	EthSwtStreamIdentificationEntry
Upstream mapping defined	[14, CP_TPS_SystemTemplate]

▽

△

Description	Configuration of a filter rule. **Tags:** atp.Status=draft
Multiplicity	1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
EthSwtStreamFilterEtherType	0..1	[ECUC_EthSwt_00170]
EthSwtStreamFilterVlanId	0..1	[ECUC_EthSwt_00168]
EthSwtStreamFilterVlanPriority	0..1	[ECUC_EthSwt_00169]

Included Containers		
Container Name	**Multiplicity**	**Description**
EthSwtStreamFilterIEEE1722 StreamId	0..1	Definition of the filter IEEE1722 Stream Id. Specifies a 64 bit Stream Id. Please note: According to IEEE802.1Q specification, the 48 most significant bits represents a source MAC address and the 16 least significant bits a unique id. AUTOSAR deviates from this definition and allow to have any kind of MAC address (e.g. source or destination or MAC multicast address)
EthSwtStreamFilterIPDestAddress	0..1	Configuration of one IP destination filter. **Tags:** atp.Status=draft
EthSwtStreamFilterIPSrcAddress	0..1	Configuration of one IP source filter. **Tags:** atp.Status=draft
EthSwtStreamFilterMACDest Address	0..1	Configuration of one MAC destination filter. **Tags:** atp.Status=draft
EthSwtStreamFilterMACSrc Address	0..1	Configuration of one MAC source filter. **Tags:** atp.Status=draft
EthSwtStreamFilterTcpDestPort	0..1	Configuration of a TCP destination port filter. **Tags:** atp.Status=draft
EthSwtStreamFilterTcpSrcPort	0..1	Configuration of a TCP source port filter. **Tags:** atp.Status=draft
EthSwtStreamFilterUdpDestPort	0..1	Configuration of a UDP destination port filter. **Tags:** atp.Status=draft
EthSwtStreamFilterUdpSrcPort	0..1	Configuration of a UDP source port filter. **Tags:** atp.Status=draft

⌋

[ECUC_EthSwt_00170] Definition of EcucIntegerParamDef EthSwtStreamFilterEtherType

Status: DRAFT

⌈

Parameter Name	EthSwtStreamFilterEtherType
Parent Container	EthSwtStreamFilterRule
Upstream mapping defined	–

▽

△

Description	Definition of the filter Ether Type. **Tags:** atp.Status=draft	
Multiplicity	0..1	
Type	EcucIntegerParamDef	
Range	0 .. 65535	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_EthSwt_00168] Definition of EcucIntegerParamDef EthSwtStreamFilterVlanId

Status: DRAFT

⌈

Parameter Name	EthSwtStreamFilterVlanId	
Parent Container	EthSwtStreamFilterRule	
Upstream mapping defined	–	
Description	Definition of the filter VLAN-ID. Note: for double tagged Ethernet frames the filter rule primitive shall be applied to the C-Tag. **Tags:** atp.Status=draft	
Multiplicity	0..1	
Type	EcucIntegerParamDef	
Range	0 .. 4094	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_EthSwt_00169] Definition of EcucIntegerParamDef EthSwtStreamFilterVlanPriority

Status: DRAFT

⌈

Parameter Name	EthSwtStreamFilterVlanPriority	
Parent Container	EthSwtStreamFilterRule	
Upstream mapping defined	–	
Description	Definition of the filter VLAN Priority. Note: for double tagged Ethernet frames the filter rule primitive shall be applied to the C-Tag. **Tags:** atp.Status=draft	
Multiplicity	0..1	
Type	EcucIntegerParamDef	
Range	0 .. 7	
Default value	–	

▽

△

Pre-Compile only	true

⌋

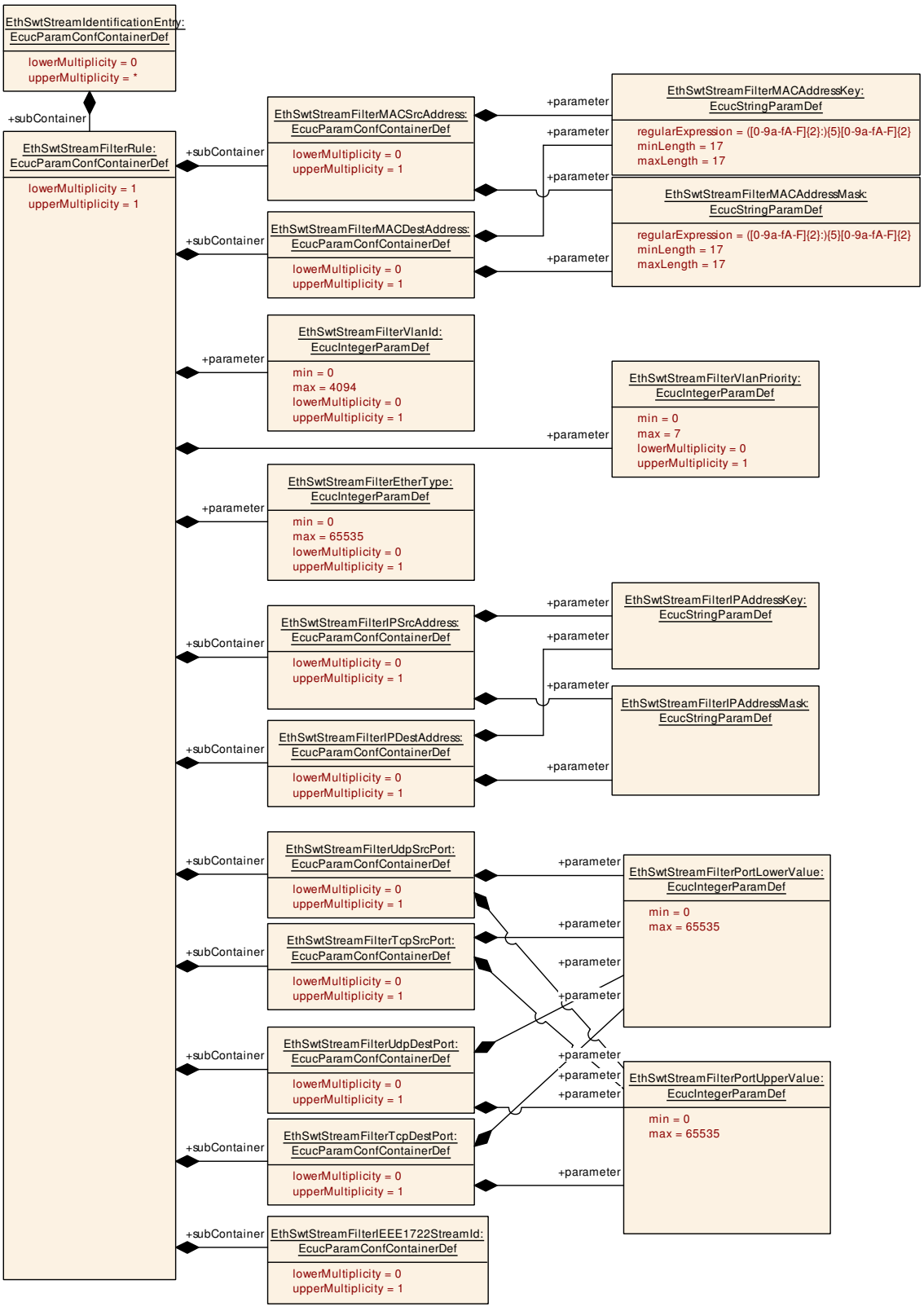

Figure 10.19: `EthSwtStreamFilterRule`

10.1.44 EthSwtStreamFilterIPDestAddress

[ECUC_EthSwt_00172] Definition of EcucParamConfContainerDef EthSwtStream FilterIPDestAddress

Status: DRAFT

⌈

Container Name	EthSwtStreamFilterIPDestAddress	
Parent Container	EthSwtStreamFilterRule	
Upstream mapping defined	–	
Description	Configuration of one IP destination filter. **Tags:** atp.Status=draft	
Multiplicity	0..1	
Post-Build Multiplicity	Loadable	Selectable
Configuration Parameters		

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
EthSwtStreamFilterIPAddressKey	1	[ECUC_EthSwt_00173]
EthSwtStreamFilterIPAddressMask	1	[ECUC_EthSwt_00174]

No Included Containers

⌋

[ECUC_EthSwt_00173] Definition of EcucStringParamDef EthSwtStreamFilter IPAddressKey

Status: DRAFT

⌈

Parameter Name	EthSwtStreamFilterIPAddressKey	
Parent Container	EthSwtStreamFilterIPDestAddress, EthSwtStreamFilterIPSrcAddress	
Upstream mapping defined	–	
Description	IP address key pattern. **Tags:** atp.Status=draft	
Multiplicity	1	
Type	EcucStringParamDef	
Default value	–	
Regular Expression	–	
Post-Build Value	Loadable	Selectable

⌋

[ECUC_EthSwt_00174] Definition of EcucStringParamDef EthSwtStreamFilterIPAddressMask

Status: DRAFT

⌈

Parameter Name	EthSwtStreamFilterIPAddressMask	
Parent Container	EthSwtStreamFilterIPDestAddress, EthSwtStreamFilterIPSrcAddress	
Upstream mapping defined	–	
Description	IP address mask pattern. **Tags:** atp.Status=draft	
Multiplicity	1	
Type	EcucStringParamDef	
Default value	–	
Regular Expression	–	
Post-Build Value	Loadable	Selectable

⌋

10.1.45 EthSwtStreamFilterIPSrcAddress

[ECUC_EthSwt_00171] Definition of EcucParamConfContainerDef EthSwtStreamFilterIPSrcAddress

Status: DRAFT

⌈

Container Name	EthSwtStreamFilterIPSrcAddress	
Parent Container	EthSwtStreamFilterRule	
Upstream mapping defined	–	
Description	Configuration of one IP source filter. **Tags:** atp.Status=draft	
Multiplicity	0..1	
Post-Build Multiplicity	Loadable	Selectable
Configuration Parameters		

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
EthSwtStreamFilterIPAddressKey	1	[ECUC_EthSwt_00173]
EthSwtStreamFilterIPAddressMask	1	[ECUC_EthSwt_00174]

No Included Containers

⌋

For parameter table [ECUC_EthSwt_00173] EthSwtStreamFilterIPAddressKey, see definition below container EthSwtStreamFilterIPDestAddress.

For parameter table [ECUC_EthSwt_00174] EthSwtStreamFilterIPAddressMask, see definition below container EthSwtStreamFilterIPDestAddress.

10.1.46 EthSwtStreamFilterMACDestAddress

[ECUC_EthSwt_00165] Definition of EcucParamConfContainerDef EthSwtStream FilterMACDestAddress

Status: DRAFT

⌈

Container Name	EthSwtStreamFilterMACDestAddress	
Parent Container	EthSwtStreamFilterRule	
Upstream mapping defined	–	
Description	Configuration of one MAC destination filter. **Tags:** atp.Status=draft	
Multiplicity	0..1	
Post-Build Multiplicity	Loadable	Selectable
Configuration Parameters		

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
EthSwtStreamFilterMACAddressKey	1	[ECUC_EthSwt_00166]
EthSwtStreamFilterMACAddressMask	1	[ECUC_EthSwt_00167]

No Included Containers

⌋

[ECUC_EthSwt_00166] Definition of EcucStringParamDef EthSwtStreamFilter MACAddressKey

Status: DRAFT

⌈

Parameter Name	EthSwtStreamFilterMACAddressKey	
Parent Container	EthSwtStreamFilterMACDestAddress, EthSwtStreamFilterMACSrcAddress	
Upstream mapping defined	–	
Description	Specifies the 48-bit physical address (MAC address) key value. **Tags:** atp.Status=draft	
Multiplicity	1	
Type	EcucStringParamDef	
Default value	–	
Length	17-17	
Regular Expression	([0-9a-fA-F]{2}:){5}[0-9a-fA-F]{2}	
Post-Build Value	Loadable	Selectable

⌋

[ECUC_EthSwt_00167] Definition of EcucStringParamDef EthSwtStreamFilterMACAddressMask

Status: DRAFT

⌈

Parameter Name	EthSwtStreamFilterMACAddressMask	
Parent Container	EthSwtStreamFilterMACDestAddress, EthSwtStreamFilterMACSrcAddress	
Upstream mapping defined	–	
Description	Specifies the 48-bit physical address (MAC address) mask value. **Tags:** atp.Status=draft	
Multiplicity	1	
Type	EcucStringParamDef	
Default value	–	
Length	17-17	
Regular Expression	([0-9a-fA-F]{2}:){5}[0-9a-fA-F]{2}	
Post-Build Value	Loadable	Selectable

⌋

10.1.47 EthSwtStreamFilterMACSrcAddress

[ECUC_EthSwt_00164] Definition of EcucParamConfContainerDef EthSwtStreamFilterMACSrcAddress

Status: DRAFT

⌈

Container Name	EthSwtStreamFilterMACSrcAddress	
Parent Container	EthSwtStreamFilterRule	
Upstream mapping defined	–	
Description	Configuration of one MAC source filter. **Tags:** atp.Status=draft	
Multiplicity	0..1	
Post-Build Multiplicity	Loadable	Selectable
Configuration Parameters		

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
EthSwtStreamFilterMACAddressKey	1	[ECUC_EthSwt_00166]
EthSwtStreamFilterMACAddressMask	1	[ECUC_EthSwt_00167]

No Included Containers

⌋

For parameter table [ECUC_EthSwt_00166] EthSwtStreamFilterMACAddressKey, see definition below container EthSwtStreamFilterMACDestAddress.

For parameter table [ECUC_EthSwt_00167] EthSwtStreamFilterMACAddressMask, see definition below container EthSwtStreamFilterMACDestAddress.

10.1.48 EthSwtStreamFilterTcpDestPort

[ECUC_EthSwt_00178] Definition of EcucParamConfContainerDef EthSwtStreamFilterTcpDestPort

Status: DRAFT

⌈

Container Name	EthSwtStreamFilterTcpDestPort	
Parent Container	EthSwtStreamFilterRule	
Upstream mapping defined	–	
Description	Configuration of a TCP destination port filter. **Tags:** atp.Status=draft	
Multiplicity	0..1	
Post-Build Multiplicity	Loadable	Selectable
Configuration Parameters		

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
EthSwtStreamFilterPortLowerValue	1	[ECUC_EthSwt_00179]
EthSwtStreamFilterPortUpperValue	1	[ECUC_EthSwt_00180]

No Included Containers

⌋

[ECUC_EthSwt_00179] Definition of EcucIntegerParamDef EthSwtStreamFilterPortLowerValue

Status: DRAFT

⌈

Parameter Name	EthSwtStreamFilterPortLowerValue	
Parent Container	EthSwtStreamFilterTcpDestPort, EthSwtStreamFilterTcpSrcPort, EthSwtStreamFilterUdpDestPort, EthSwtStreamFilterUdpSrcPort	
Upstream mapping defined	–	
Description	Definition of the filter port lower value. **Tags:** atp.Status=draft	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 65535	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_EthSwt_00180] Definition of EcucIntegerParamDef EthSwtStreamFilterPortUpperValue

Status: DRAFT

⌈

Parameter Name	EthSwtStreamFilterPortUpperValue	
Parent Container	EthSwtStreamFilterTcpDestPort, EthSwtStreamFilterTcpSrcPort, EthSwtStreamFilterUdpDestPort, EthSwtStreamFilterUdpSrcPort	
Upstream mapping defined	–	
Description	Definition of the filter port upper value. **Tags:** atp.Status=draft	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 65535	
Default value	–	
Pre-Compile only	true	

⌋

10.1.49 EthSwtStreamFilterTcpSrcPort

[ECUC_EthSwt_00176] Definition of EcucParamConfContainerDef EthSwtStreamFilterTcpSrcPort

Status: DRAFT

⌈

Container Name	EthSwtStreamFilterTcpSrcPort	
Parent Container	EthSwtStreamFilterRule	
Upstream mapping defined	–	
Description	Configuration of a TCP source port filter. **Tags:** atp.Status=draft	
Multiplicity	0..1	
Post-Build Multiplicity	Loadable	Selectable
Configuration Parameters		

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
EthSwtStreamFilterPortLowerValue	1	[ECUC_EthSwt_00179]
EthSwtStreamFilterPortUpperValue	1	[ECUC_EthSwt_00180]

No Included Containers

⌋

For parameter table [ECUC_EthSwt_00179] EthSwtStreamFilterPortLowerValue, see definition below container EthSwtStreamFilterTcpDestPort.

For parameter table [ECUC_EthSwt_00180] EthSwtStreamFilterPortUpperValue, see definition below container EthSwtStreamFilterTcpDestPort.

10.1.50 EthSwtStreamFilterUdpDestPort

[ECUC_EthSwt_00177] Definition of EcucParamConfContainerDef EthSwtStreamFilterUdpDestPort

Status: DRAFT

⌈

Container Name	EthSwtStreamFilterUdpDestPort	
Parent Container	EthSwtStreamFilterRule	
Upstream mapping defined	–	
Description	Configuration of a UDP destination port filter. **Tags:** atp.Status=draft	
Multiplicity	0..1	
Post-Build Multiplicity	Loadable	Selectable
Configuration Parameters		

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
EthSwtStreamFilterPortLowerValue	1	[ECUC_EthSwt_00179]
EthSwtStreamFilterPortUpperValue	1	[ECUC_EthSwt_00180]

No Included Containers

⌋

For parameter table [ECUC_EthSwt_00179] EthSwtStreamFilterPortLowerValue, see definition below container EthSwtStreamFilterTcpDestPort.

For parameter table [ECUC_EthSwt_00180] EthSwtStreamFilterPortUpperValue, see definition below container EthSwtStreamFilterTcpDestPort.

10.1.51 EthSwtStreamFilterUdpSrcPort

[ECUC_EthSwt_00175] Definition of EcucParamConfContainerDef EthSwtStreamFilterUdpSrcPort

Status: DRAFT

⌈

Container Name	EthSwtStreamFilterUdpSrcPort	
Parent Container	EthSwtStreamFilterRule	
Upstream mapping defined	–	
Description	Configuration of a UDP source port filter. **Tags:** atp.Status=draft	
Multiplicity	0..1	
Post-Build Multiplicity	Loadable	Selectable
Configuration Parameters		

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
EthSwtStreamFilterPortLowerValue	1	[ECUC_EthSwt_00179]
EthSwtStreamFilterPortUpperValue	1	[ECUC_EthSwt_00180]

No Included Containers

⌋

For parameter table [ECUC_EthSwt_00179] EthSwtStreamFilterPortLowerValue, see definition below container EthSwtStreamFilterTcpDestPort.

For parameter table [ECUC_EthSwt_00180] EthSwtStreamFilterPortUpperValue, see definition below container EthSwtStreamFilterTcpDestPort.

10.1.52 EthSwtStreamFilterIEEE1722StreamId

[ECUC_EthSwt_00232] Definition of EcucParamConfContainerDef EthSwtStreamFilterIEEE1722StreamId ⌈

Container Name	EthSwtStreamFilterIEEE1722StreamId	
Parent Container	EthSwtStreamFilterRule	
Upstream mapping defined	[14, CP_TPS_SystemTemplate]	
Description	Definition of the filter IEEE1722 Stream Id. Specifies a 64 bit Stream Id. Please note: According to IEEE802.1Q specification, the 48 most significant bits represents a source MAC address and the 16 least significant bits a unique id. AUTOSAR deviates from this definition and allow to have any kind of MAC address (e.g. source or destination or MAC multicast address)	
Multiplicity	0..1	
Post-Build Multiplicity	-	Selectable
Configuration Parameters		

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
EthSwtStreamFilterStreamMacId	1	[ECUC_EthSwt_00233]
EthSwtStreamFilterStreamUniqueId	1	[ECUC_EthSwt_00234]

No Included Containers

⌋

[ECUC_EthSwt_00233] Definition of EcucStringParamDef EthSwtStreamFilterStreamMacId ⌈

Parameter Name	EthSwtStreamFilterStreamMacId	
Parent Container	EthSwtStreamFilterIEEE1722StreamId	
Upstream mapping defined	[14, CP_TPS_SystemTemplate]	
Description	Specifies the 48-bit physical address (MAC address) part of the IEEE1722 Stream Id.	
Multiplicity	1	
Type	EcucStringParamDef	
Default value	–	
Length	17-17	
Regular Expression	([0-9a-fA-F]{2}:){5}[0-9a-fA-F]{2}	
Post-Build Value	-	Selectable

⌋

[ECUC_EthSwt_00234] Definition of EcucIntegerParamDef EthSwtStreamFilterStreamUniqueId ⌈

Parameter Name	EthSwtStreamFilterStreamUniqueId	
Parent Container	EthSwtStreamFilterIEEE1722StreamId	
Upstream mapping defined	[14, CP_TPS_SystemTemplate]	
Description	Specifies the 16-bit unique part of the IEEE1722 Stream Id.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 65535	
Default value	0	
Post-Build Value	-	Selectable

⌋

10.1.53 EthSwtUnknownMacDestAddressConfig

[ECUC_EthSwt_00239] Definition of EcucParamConfContainerDef EthSwtUnknownMacDestAddressConfig ⌈

Container Name	EthSwtUnknownMacDestAddressConfig	
Parent Container	EthSwtConfig	
Upstream mapping defined	[14, CP_TPS_SystemTemplate]	
Description	Definition to which EthSwtPorts an Ethernet frame shall be forwarded if the destination MAC address is not present in the address resolution lookup (ARL) table.	
Multiplicity	1..*	
Post-Build Multiplicity	Loadable	Selectable
Configuration Parameters		

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
EthSwtDestPortsForUnknownMulticastMacDestAddressRef	0..255	[ECUC_EthSwt_00241]
EthSwtDestPortsForUnknownUnicastMacDestAddressRef	0..255	[ECUC_EthSwt_00240]
EthSwtDestVlanForUnknownMacDestAddressRef	0..4095	[ECUC_EthSwt_00242]

No Included Containers

⌋

[ECUC_EthSwt_00241] Definition of EcucReferenceDef EthSwtDestPortsForUnknownMulticastMacDestAddressRef ⌈

Parameter Name	EthSwtDestPortsForUnknownMulticastMacDestAddressRef	
Parent Container	EthSwtUnknownMacDestAddressConfig	
Upstream mapping defined	[14, CP_TPS_SystemTemplate]	
Description	This parameter specifies the egress ports frames with unknown multicast MAC destination addresses (without a matching ARL entry) are forwarded on. Note that "Drop Unknown Multicast" behavior can be achieved by not referencing any EthSwtPort.	
Multiplicity	0..255	
Type	Reference to EthSwtPort	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

⌋

[ECUC_EthSwt_00240] Definition of EcucReferenceDef EthSwtDestPortsForUnknownUnicastMacDestAddressRef ⌈

Parameter Name	EthSwtDestPortsForUnknownUnicastMacDestAddressRef	
Parent Container	EthSwtUnknownMacDestAddressConfig	
Upstream mapping defined	[14, CP_TPS_SystemTemplate]	
Description	This parameter specifies the egress ports frames with unknown unicast MAC destination addresses (without a matching ARL entry) are forwarded on. Note that "Flooding" can be achieved by referencing all EthSwtPorts.	
Multiplicity	0..255	
Type	Reference to EthSwtPort	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

⌋

[ECUC_EthSwt_00242] Definition of EcucReferenceDef EthSwtDestVlanForUnknownMacDestAddressRef ⌈

Parameter Name	EthSwtDestVlanForUnknownMacDestAddressRef
Parent Container	EthSwtUnknownMacDestAddressConfig
Upstream mapping defined	[14, CP_TPS_SystemTemplate]

▽

△

Description	Optional reference to a set of VLANs to define that the owning EthSwtUnknownMac DestAddressConfig is applicable for these VLANs only.	
Multiplicity	0..4095	
Type	Reference to EthSwtVlanMembership	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

⌋

10.1.54 EthSwtVlanMembership

[ECUC_EthSwt_00199] Definition of EcucParamConfContainerDef EthSwtVlanMembership ⌈

Container Name	EthSwtVlanMembership	
Parent Container	EthSwtConfig	
Upstream mapping defined	[14, CP_TPS_SystemTemplate]	
Description	Determines the membership of this Ethernet switch and the referenced ports to the virtual network, i.e. frames with this VID can be received and transmitted via the referenced ports.	
Multiplicity	0..4095	
Post-Build Multiplicity	Loadable	Selectable
Configuration Parameters		

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
EthSwtVlanMembershipId	1	[ECUC_EthSwt_00202]

Included Containers		
Container Name	**Multiplicity**	**Description**
EthSwtVlanMembershipPortRefEntry	0..255	Determines the VLAN membership of one referenced ports to the virtual network and the according forwarding type (NOT_SENT, SENT_UNTAGGED, SENT_TAGGED).

⌋

[ECUC_EthSwt_00202] Definition of EcucIntegerParamDef EthSwtVlanMembershipId ⌈

Parameter Name	EthSwtVlanMembershipId	
Parent Container	EthSwtVlanMembership	
Upstream mapping defined	[14, CP_TPS_SystemTemplate]	
Description	Determines the VID of the virtual network this port belongs to.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 4094	

▽

△

Default value	–	
Post-Build Value	Loadable	Selectable

⌋

10.1.55 EthSwtVlanMembershipPortRefEntry

[ECUC_EthSwt_00203] Definition of EcucParamConfContainerDef EthSwtVlanMembershipPortRefEntry ⌈

Container Name	EthSwtVlanMembershipPortRefEntry	
Parent Container	EthSwtVlanMembership	
Upstream mapping defined	[14, CP_TPS_SystemTemplate]	
Description	Determines the VLAN membership of one referenced ports to the virtual network and the according forwarding type (NOT_SENT, SENT_UNTAGGED, SENT_TAGGED).	
Multiplicity	0..255	
Post-Build Multiplicity	Loadable	Selectable
Configuration Parameters		

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
EthSwtVlanForwardingType	1	[ECUC_EthSwt_00026]
EthSwtVlanMembershipPortRef	1	[ECUC_EthSwt_00204]

No Included Containers

⌋

[ECUC_EthSwt_00026] Definition of EcucEnumerationParamDef EthSwtVlanForwardingType ⌈

Parameter Name	EthSwtVlanForwardingType	
Parent Container	EthSwtVlanMembershipPortRefEntry	
Upstream mapping defined	[14, CP_TPS_SystemTemplate]	
Description	Defines how the message with a specific VLAN-ID at the referenced port shall be handled.	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	ETHSWT_NOT_SENT	The message with the specific VLAN Id shall not be sent at the referenced port.
	ETHSWT_SENT_TAGGED	The message with the specific VLAN Id shall be sent with its VLAN Id at the referenced port.
	ETHSWT_SENT_UNTAGGED	The message with the specific VLAN Id shall be sent untagged at the referenced port.
Post-Build Value	Loadable	Selectable

⌋

[ECUC_EthSwt_00204] Definition of EcucReferenceDef EthSwtVlanMembership PortRef ⌈

Parameter Name	EthSwtVlanMembershipPortRef	
Parent Container	EthSwtVlanMembershipPortRefEntry	
Upstream mapping defined	[14, CP_TPS_SystemTemplate]	
Description	Reference to one port the VLAN shall be assigned to.	
Multiplicity	1	
Type	Reference to EthSwtPort	
Post-Build Value	Loadable	Selectable

⌋

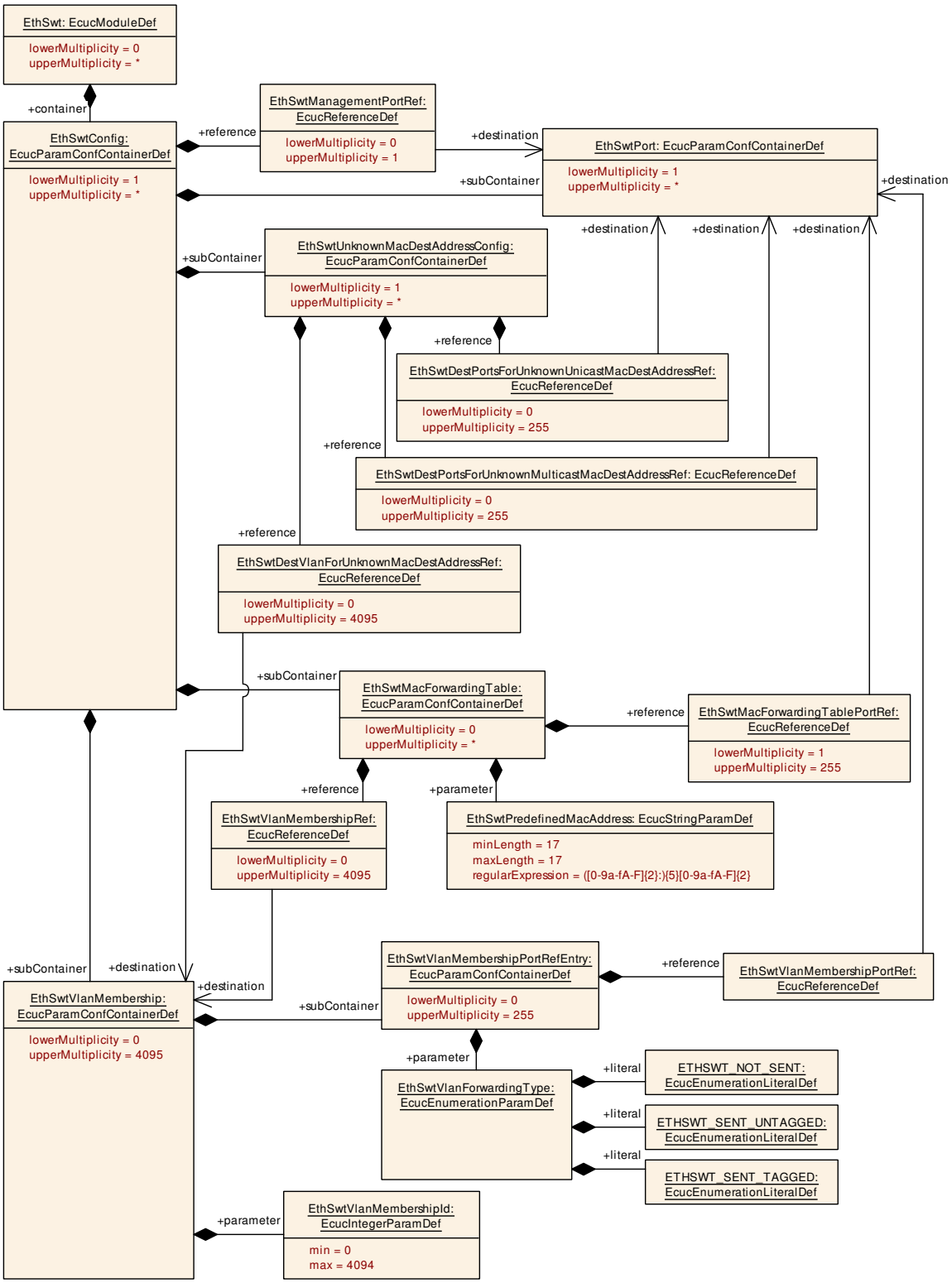

Figure 10.20: `EthSwtVlanMembershipPortRefEntry`

10.2 Constraints

[SWS_EthSwt_CONSTR_00411] ⌈The ECUC partitions referenced by EthSwtConfigEcucPartitionRef shall be a subset of the ECUC partitions referenced by EthSwtEcucPartitionRef.⌋

[SWS_EthSwt_CONSTR_00412] ⌈EthSwtConfig, EthCtrlConfig and EthTrcvConfig of one communication channel shall all reference the same ECUC partition.⌋

[SWS_EthSwt_CONSTR_00438] ⌈If EthSwtEcucPartitionRef references one or more ECUC partitions, EthSwtConfigEcucPartitionRef shall have a multiplicity of one and reference one of these ECUC partitions as well.⌋

A 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.

A.1 Traceable item history of this document according to AUTOSAR Release R22-11

A.1.1 Added Specification Items in R22-11

Number	Heading
[SWS_EthSwt_00455]	
[SWS_EthSwt_00460]	
[SWS_EthSwt_00461]	
[SWS_EthSwt_00462]	
[SWS_EthSwt_00463]	
[SWS_EthSwt_00465]	
[SWS_EthSwt_00466]	
[SWS_EthSwt_00467]	
[SWS_EthSwt_00469]	
[SWS_EthSwt_00471]	
[SWS_EthSwt_00472]	
[SWS_EthSwt_00475]	
[SWS_EthSwt_00476]	
[SWS_EthSwt_00477]	
[SWS_EthSwt_00478]	
[SWS_EthSwt_00479]	
[SWS_EthSwt_00480]	
[SWS_EthSwt_00481]	
[SWS_EthSwt_00482]	
[SWS_EthSwt_00483]	
[SWS_EthSwt_00484]	
[SWS_EthSwt_00486]	
[SWS_EthSwt_00487]	
[SWS_EthSwt_00490]	
[SWS_EthSwt_00491]	
[SWS_EthSwt_00492]	
[SWS_EthSwt_00493]	



△

Number	Heading
[SWS_EthSwt_00494]	
[SWS_EthSwt_91124]	
[SWS_EthSwt_91125]	
[SWS_EthSwt_91126]	
[SWS_EthSwt_91127]	
[SWS_EthSwt_91128]	
[SWS_EthSwt_91129]	
[SWS_EthSwt_91130]	
[SWS_EthSwt_91131]	
[SWS_EthSwt_91132]	
[SWS_EthSwt_91133]	
[SWS_EthSwt_91134]	
[SWS_EthSwt_91135]	
[SWS_EthSwt_91136]	
[SWS_EthSwt_91137]	
[SWS_EthSwt_91138]	
[SWS_EthSwt_91139]	
[SWS_EthSwt_ CONSTR_00450]	
[SWS_EthSwt_ CONSTR_00451]	
[SWS_EthSwt_ CONSTR_00452]	
[SWS_EthSwt_ CONSTR_00453]	
[SWS_EthSwt_ CONSTR_00454]	
[SWS_EthSwt_ CONSTR_00456]	
[SWS_EthSwt_ CONSTR_00457]	
[SWS_EthSwt_ CONSTR_00458]	
[SWS_EthSwt_ CONSTR_00459]	
[SWS_EthSwt_ CONSTR_00464]	
[SWS_EthSwt_ CONSTR_00468]	
[SWS_EthSwt_ CONSTR_00470]	

▽

△

Number	Heading
[SWS_EthSwt_ CONSTR_00473]	
[SWS_EthSwt_ CONSTR_00474]	
[SWS_EthSwt_ CONSTR_00485]	
[SWS_EthSwt_ CONSTR_00488]	
[SWS_EthSwt_ CONSTR_00489]	
[SWS_EthSwt_ CONSTR_00495]	
[SWS_EthSwt_ CONSTR_00496]	

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

A.1.2 Changed Specification Items in R22-11

Number	Heading
[SWS_EthSwt_00001]	
[SWS_EthSwt_00002]	
[SWS_EthSwt_00006]	
[SWS_EthSwt_00009]	
[SWS_EthSwt_00018]	
[SWS_EthSwt_00019]	
[SWS_EthSwt_00023]	
[SWS_EthSwt_00025]	
[SWS_EthSwt_00026]	
[SWS_EthSwt_00031]	
[SWS_EthSwt_00032]	
[SWS_EthSwt_00037]	
[SWS_EthSwt_00038]	
[SWS_EthSwt_00044]	
[SWS_EthSwt_00045]	
[SWS_EthSwt_00051]	
[SWS_EthSwt_00052]	
[SWS_EthSwt_00058]	
[SWS_EthSwt_00060]	

▽

△

Number	Heading
[SWS_EthSwt_00061]	
[SWS_EthSwt_00086]	
[SWS_EthSwt_00087]	
[SWS_EthSwt_00091]	
[SWS_EthSwt_00092]	
[SWS_EthSwt_00098]	
[SWS_EthSwt_00106]	
[SWS_EthSwt_00111]	
[SWS_EthSwt_00114]	
[SWS_EthSwt_00117]	
[SWS_EthSwt_00118]	
[SWS_EthSwt_00123]	
[SWS_EthSwt_00125]	
[SWS_EthSwt_00126]	
[SWS_EthSwt_00127]	
[SWS_EthSwt_00128]	
[SWS_EthSwt_00132]	
[SWS_EthSwt_00133]	
[SWS_EthSwt_00134]	
[SWS_EthSwt_00135]	
[SWS_EthSwt_00136]	
[SWS_EthSwt_00154]	
[SWS_EthSwt_00156]	
[SWS_EthSwt_00157]	
[SWS_EthSwt_00162]	
[SWS_EthSwt_00164]	
[SWS_EthSwt_00165]	
[SWS_EthSwt_00172]	
[SWS_EthSwt_00173]	
[SWS_EthSwt_00178]	
[SWS_EthSwt_00179]	
[SWS_EthSwt_00180]	
[SWS_EthSwt_00181]	
[SWS_EthSwt_00182]	
[SWS_EthSwt_00183]	
[SWS_EthSwt_00187]	
[SWS_EthSwt_00188]	
[SWS_EthSwt_00193]	

▽

△

Number	Heading
[SWS_EthSwt_00194]	
[SWS_EthSwt_00196]	
[SWS_EthSwt_00197]	
[SWS_EthSwt_00198]	
[SWS_EthSwt_00199]	
[SWS_EthSwt_00203]	
[SWS_EthSwt_00204]	
[SWS_EthSwt_00206]	
[SWS_EthSwt_00211]	
[SWS_EthSwt_00212]	
[SWS_EthSwt_00216]	
[SWS_EthSwt_00217]	
[SWS_EthSwt_00221]	
[SWS_EthSwt_00222]	
[SWS_EthSwt_00226]	
[SWS_EthSwt_00227]	
[SWS_EthSwt_00228]	
[SWS_EthSwt_00231]	
[SWS_EthSwt_00233]	
[SWS_EthSwt_00234]	
[SWS_EthSwt_00235]	
[SWS_EthSwt_00240]	
[SWS_EthSwt_00241]	
[SWS_EthSwt_00242]	
[SWS_EthSwt_00243]	
[SWS_EthSwt_00245]	
[SWS_EthSwt_00372]	
[SWS_EthSwt_00373]	
[SWS_EthSwt_00378]	
[SWS_EthSwt_00398]	
[SWS_EthSwt_00416]	
[SWS_EthSwt_00417]	
[SWS_EthSwt_00418]	
[SWS_EthSwt_00419]	
[SWS_EthSwt_00420]	
[SWS_EthSwt_00430]	
[SWS_EthSwt_00431]	
[SWS_EthSwt_00434]	

▽

△

Number	Heading
[SWS_EthSwt_00440]	
[SWS_EthSwt_91000]	
[SWS_EthSwt_91001]	
[SWS_EthSwt_91002]	
[SWS_EthSwt_91003]	
[SWS_EthSwt_91004]	
[SWS_EthSwt_91005]	
[SWS_EthSwt_91006]	
[SWS_EthSwt_91007]	
[SWS_EthSwt_91008]	
[SWS_EthSwt_91009]	
[SWS_EthSwt_91010]	
[SWS_EthSwt_91011]	
[SWS_EthSwt_91012]	
[SWS_EthSwt_91013]	
[SWS_EthSwt_91014]	
[SWS_EthSwt_91015]	
[SWS_EthSwt_91016]	
[SWS_EthSwt_91017]	
[SWS_EthSwt_91018]	
[SWS_EthSwt_91019]	
[SWS_EthSwt_91020]	
[SWS_EthSwt_91021]	
[SWS_EthSwt_91022]	
[SWS_EthSwt_91023]	
[SWS_EthSwt_91024]	
[SWS_EthSwt_91025]	
[SWS_EthSwt_91028]	
[SWS_EthSwt_91029]	
[SWS_EthSwt_91030]	
[SWS_EthSwt_91031]	
[SWS_EthSwt_91032]	
[SWS_EthSwt_91033]	
[SWS_EthSwt_91034]	
[SWS_EthSwt_91035]	
[SWS_EthSwt_91036]	
[SWS_EthSwt_91037]	
[SWS_EthSwt_91038]	

▽

△

Number	Heading
[SWS_EthSwt_91039]	
[SWS_EthSwt_91040]	
[SWS_EthSwt_91050]	
[SWS_EthSwt_91104]	
[SWS_EthSwt_91123]	

Table A.2: Changed Specification Items in R22-11

A.1.3 Deleted Specification Items in R22-11

none

A.1.4 Added Constraints in R22-11

none

A.1.5 Changed Constraints in R22-11

none

A.1.6 Deleted Constraints in R22-11

none

A.2 Traceable item history of this document according to AUTOSAR Release R23-11

A.2.1 Added Specification Items in R23-11

Number	Heading
[SWS_EthSwt_00163]	
[SWS_EthSwt_00450]	
[SWS_EthSwt_00451]	
[SWS_EthSwt_00456]	
[SWS_EthSwt_00459]	

▽

△

Number	Heading
[SWS_EthSwt_00500]	
[SWS_EthSwt_00501]	
[SWS_EthSwt_00502]	
[SWS_EthSwt_00503]	
[SWS_EthSwt_00504]	
[SWS_EthSwt_00601]	
[SWS_EthSwt_00602]	
[SWS_EthSwt_00604]	
[SWS_EthSwt_00605]	
[SWS_EthSwt_00606]	
[SWS_EthSwt_00607]	
[SWS_EthSwt_00608]	
[SWS_EthSwt_00609]	
[SWS_EthSwt_00610]	
[SWS_EthSwt_00611]	
[SWS_EthSwt_00612]	
[SWS_EthSwt_00613]	
[SWS_EthSwt_00614]	
[SWS_EthSwt_91041]	Definition of API function EthSwt_SetStreamHandleIdxConfiguration
[SWS_EthSwt_91042]	Definition of API function EthSwt_GetStreamHandleIdxStatistics
[SWS_EthSwt_91043]	Definition of API function EthSwt_ExtractStreamHandleIdx

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

A.2.2 Changed Specification Items in R23-11

Number	Heading
[SWS_EthSwt_00002]	Definition of imported datatypes of module EthSwt
[SWS_EthSwt_00016]	
[SWS_EthSwt_00133]	
[SWS_EthSwt_00179]	
[SWS_EthSwt_00455]	
[SWS_EthSwt_00465]	
[SWS_EthSwt_00467]	
[SWS_EthSwt_00469]	
[SWS_EthSwt_00471]	
[SWS_EthSwt_00472]	

▽

△

Number	Heading
[SWS_EthSwt_00475]	
[SWS_EthSwt_00476]	
[SWS_EthSwt_00478]	
[SWS_EthSwt_00493]	

Table A.4: Changed Specification Items in R23-11

A.2.3 Deleted Specification Items in R23-11

Number	Heading
[SWS_EthSwt_00136]	
[SWS_EthSwt_00162]	
[SWS_EthSwt_00181]	
[SWS_EthSwt_00466]	
[SWS_EthSwt_00490]	

Table A.5: Deleted Specification Items in R23-11

A.2.4 Added Constraints in R23-11

Number	Heading
[SWS_EthSwt_CONSTR_00602]	
[SWS_EthSwt_CONSTR_00603]	

Table A.6: Added Constraints in R23-11

A.2.5 Changed Constraints in R23-11

Number	Heading
[SWS_EthSwt_CONSTR_00453]	
[SWS_EthSwt_CONSTR_00454]	
[SWS_EthSwt_CONSTR_00464]	
[SWS_EthSwt_CONSTR_00468]	

Table A.7: Changed Constraints in R23-11

A.2.6 Deleted Constraints in R23-11

Number	Heading
[SWS_EthSwt_CONSTR_00450]	
[SWS_EthSwt_CONSTR_00451]	
[SWS_EthSwt_CONSTR_00456]	
[SWS_EthSwt_CONSTR_00458]	
[SWS_EthSwt_CONSTR_00459]	

▽

△

Number	Heading
[SWS_EthSwt_CONSTR_00473]	
[SWS_EthSwt_CONSTR_00474]	
[SWS_EthSwt_CONSTR_00488]	
[SWS_EthSwt_CONSTR_00496]	

Table A.8: Deleted Constraints in R23-11

A.3 Traceable item history of this document according to AUTOSAR Release R24-11

A.3.1 Added Specification Items in R24-11

Number	Heading
[ECUC_EthSwt_00232]	Definition of EcucParamConfContainerDef EthSwtStreamFilter IEEE1722StreamId
[ECUC_EthSwt_00233]	Definition of EcucStringParamDef EthSwtStreamFilterStreamMacId
[ECUC_EthSwt_00234]	Definition of EcucIntegerParamDef EthSwtStreamFilterStreamUniqueId
[ECUC_EthSwt_00235]	Definition of EcucBooleanParamDef EthSwtGetPortMacAddrVlanApi
[ECUC_EthSwt_00236]	Definition of EcucEnumerationParamDef EthSwtMacAddressLearningMode
[ECUC_EthSwt_00237]	Definition of EcucReferenceDef EthSwtVlanMembershipRef
[ECUC_EthSwt_00238]	Definition of EcucIntegerParamDef EthSwtPortInterPacketGap
[ECUC_EthSwt_00239]	Definition of EcucParamConfContainerDef EthSwtUnknownMacDest AddressConfig
[ECUC_EthSwt_00240]	Definition of EcucReferenceDef EthSwtDestPortsForUnknownUnicastMac DestAddressRef

▽

△

Number	Heading
[ECUC_EthSwt_00241]	Definition of EcucReferenceDef EthSwtDestPortsForUnknownMulticastMacDestAddressRef
[ECUC_EthSwt_00242]	Definition of EcucReferenceDef EthSwtDestVlanForUnknownMacDestAddressRef
[ECUC_EthSwt_00243]	Definition of EcucBooleanParamDef EthSwtReadMmdApi
[ECUC_EthSwt_00244]	Definition of EcucBooleanParamDef EthSwtWriteMmdApi
[ECUC_EthSwt_00245]	Definition of EcucIntegerParamDef EthSwtUsedInternalPriorityUpperValue
[ECUC_EthSwt_00246]	Definition of EcucIntegerParamDef EthSwtUsedTrafficClassUpperValue
[ECUC_EthSwt_00247]	Definition of EcucIntegerParamDef EthSwtPortDefaultTrafficClass
[ECUC_EthSwt_00248]	Definition of EcucParamConfContainerDef EthSwtPortPriorityToTrafficClassAssignment
[ECUC_EthSwt_00249]	Definition of EcucIntegerParamDef EthSwtPortPriorityToTrafficClassAssignmentPriority
[ECUC_EthSwt_00250]	Definition of EcucIntegerParamDef EthSwtPortPriorityToTrafficClassAssignmentTrafficClass
[ECUC_EthSwt_00251]	Definition of EcucParamConfContainerDef EthSwtPortEgressQueueTransmissionSelectionETSConfig
[ECUC_EthSwt_00252]	Definition of EcucIntegerParamDef EthSwtETSConfigAvailableBandwidthInPercent
[ECUC_EthSwt_00253]	Definition of EcucIntegerParamDef EthSwtETSConfigAvailableBandwidthInWeightValue
[ECUC_EthSwt_00254]	Definition of EcucBooleanParamDef EthSwtFramePreemptionEnable
[ECUC_EthSwt_00255]	Definition of EcucEnumerationParamDef EthSwtTrafficClassToPreemptionStatusAssignment
[ECUC_EthSwt_00256]	Definition of EcucParamConfContainerDef EthSwtPortIngressVlanTranslationTable
[ECUC_EthSwt_00257]	Definition of EcucParamConfContainerDef EthSwtPortIngressVlanTranslationTableEntry
[ECUC_EthSwt_00258]	Definition of EcucIntegerParamDef EthSwtIngressVlanId
[ECUC_EthSwt_00259]	Definition of EcucIntegerParamDef EthSwtTranslatedVlanId
[SWS_EthSwt_00444]	SVL MAC address learning mode
[SWS_EthSwt_00445]	IVL MAC address learning mode
[SWS_EthSwt_00448]	Behaviour if `EthSwtMacAddressLearningMode` is set to `IVL`
[SWS_EthSwt_00449]	Behaviour if given MAC-address is available at exactly one Ethernet port
[SWS_EthSwt_00511]	Behaviour if given MAC-address is available at multiple Ethernet ports

▽

△

Number	Heading
[SWS_EthSwt_00512]	Behaviour if `EthSwtMacAddressLearningMode` is set to `SVL`
[SWS_EthSwt_00513]	Behaviour if `EthSwtMacAddressLearningMode` is set to `IVL` and VLAN-ID is set to value in a range from 1 to 4095
[SWS_EthSwt_00514]	Behaviour if `EthSwtMacAddressLearningMode` is set to `IVL` and VLAN-ID is set to 0
[SWS_EthSwt_00515]	Behaviour if `EthSwtMacAddressLearningMode` is set to `IVL` and VLAN-ID is set to a value greater than 4095
[SWS_EthSwt_00516]	Creation of Ethernet switch port bit map
[SWS_EthSwt_00517]	Behaviour for creation of Ethernet switch port bit map
[SWS_EthSwt_00518]	Behaviour if Ethernet switch port bit map exceeds uint32 data type
[SWS_EthSwt_00519]	compile configuration for API
[SWS_EthSwt_00520]	Broadcast destination
[SWS_EthSwt_00521]	Unicast destination
[SWS_EthSwt_00522]	Multicast destination
[SWS_EthSwt_00523]	Processing or dropping
[SWS_EthSwt_00524]	Triggering stream statistics readout
[SWS_EthSwt_00525]	Concatenating stream statistics for buckets
[SWS_EthSwt_00526]	Indicating the availability of stream statistics
[SWS_EthSwt_00531]	IPV as priority
[SWS_EthSwt_00535]	Determine traffic class for an Ethernet frame
[SWS_EthSwt_00537]	Enqueue Ethernet frames to the matching egress port queue
[SWS_EthSwt_00539]	Scheduling with strict priority
[SWS_EthSwt_00540]	Scheduling with enhandced traffic shaping
[SWS_EthSwt_00541]	Dequeueing Ethernet frames with strict priority scheduling
[SWS_EthSwt_00542]	Emission opportunity suspension with strict priority scheduling
[SWS_EthSwt_00543]	Emission opportunity suspension with enhanced traffic shaping
[SWS_EthSwt_00547]	Determination of egress port queue emission with available bandwidth configured in weights as amount of Ethernet frames
[SWS_EthSwt_00548]	Amount of Ethernet frames within one emission portion if available bandwidth is configured in weights as amount of Ethernet frames
[SWS_EthSwt_00549]	Tolerance of egress port queue emission within the defined measurement interval if available bandwidth is configured in percent is used
[SWS_EthSwt_00550]	Definition of bit time
[SWS_EthSwt_00551]	Determination of egress port queue emission with available bandwidth configured in percent
[SWS_EthSwt_00553]	Utilization of all neighboring egress port queues with a resumed emission opportunity
[SWS_EthSwt_00554]	Use regenerated priority value as VLAN-priority
[SWS_EthSwt_00555]	Translating VLAN IDs
[SWS_EthSwt_00556]	Wildcard for VLAN ID translation

▽

△

Number	Heading
[SWS_EthSwt_00557]	Only one wildcard entry allowed
[SWS_EthSwt_00558]	Wildcard match operation
[SWS_EthSwt_00560]	EthSwt_ReadMmd functionality
[SWS_EthSwt_00561]	Configuring EthSwt_ReadMmd availability
[SWS_EthSwt_00562]	EthSwt_WriteMmd functionality
[SWS_EthSwt_00563]	Configuring EthSwt_WriteMmd availability
[SWS_EthSwt_91051]	Definition of API function EthSwt_GetPortMacAddrVlan
[SWS_EthSwt_91052]	Definition of API function EthSwt_ReadMmd
[SWS_EthSwt_91053]	Definition of API function EthSwt_WriteMmd

Table A.9: Added Specification Items in R24-11

A.3.2 Changed Specification Items in R24-11

Number	Heading
[ECUC_EthSwt_00001]	Definition of EcucParamConfContainerDef EthSwtConfig
[ECUC_EthSwt_00003]	Definition of EcucParamConfContainerDef EthSwtGeneral
[ECUC_EthSwt_00005]	Definition of EcucParamConfContainerDef EthSwtPort
[ECUC_EthSwt_00007]	Definition of EcucParamConfContainerDef EthSwtPortEgress
[ECUC_EthSwt_00014]	Definition of EcucParamConfContainerDef EthSwtPortIngress
[ECUC_EthSwt_00018]	Definition of EcucEnumerationParamDef EthSwtPortSchedulerAlgorithm
[ECUC_EthSwt_00054]	Definition of EcucEnumerationParamDef EthSwtPortPhysicalLayerType
[ECUC_EthSwt_00114]	Definition of EcucEnumerationParamDef EthSwtPortMacLayerSpeed
[ECUC_EthSwt_00141]	Definition of EcucParamConfContainerDef EthSwtStreamFilterRule
[ECUC_EthSwt_00182]	Definition of EcucParamConfContainerDef EthSwtPortQueue
[ECUC_EthSwt_00185]	Definition of EcucIntegerParamDef EthSwtPortQueueTrafficClass Assignment
[ECUC_EthSwt_00186]	Definition of EcucParamConfContainerDef EthSwtPortEgressQueue TransmissionSelection
[ECUC_EthSwt_00205]	Definition of EcucParamConfContainerDef EthSwtMacForwardingTable

▽

△

Number	Heading
[SWS_EthSwt_00098]	Definition of optional interfaces requested by module EthSwt
[SWS_EthSwt_00502]	
[SWS_EthSwt_00602]	
[SWS_EthSwt_00604]	
[SWS_EthSwt_91134]	Definition of API function EthSwt_MacSecGetMacSecStatistics
[SWS_EthSwt_91135]	Definition of callback function EthSwt_MacSecUpdateSecYNotification
[SWS_EthSwt_91136]	Definition of callback function EthSwt_MacSecAddTxSaNotification
[SWS_EthSwt_91137]	Definition of callback function EthSwt_MacSecAddRxSaNotification
[SWS_EthSwt_91138]	Definition of callback function EthSwt_MacSecGetMacSecStatistics Notification

Table A.10: Changed Specification Items in R24-11

A.3.3 Deleted Specification Items in R24-11

Number	Heading
[ECUC_EthSwt_00015]	Definition of EcucIntegerParamDef EthSwtPortIngressVlanModification
[ECUC_EthSwt_00023]	Definition of EcucIntegerParamDef EthSwtPortTrafficClassAssignment
[ECUC_EthSwt_00027]	Definition of EcucParamConfContainerDef EthSwtPortPriorityTrafficClassAssignment
[ECUC_EthSwt_00028]	Definition of EcucIntegerParamDef EthSwtPortPriorityTrafficClassAssignmentRegeneratedPriority
[ECUC_EthSwt_00029]	Definition of EcucIntegerParamDef EthSwtPortPriorityTrafficClassAssignmentTrafficClass
[ECUC_EthSwt_00138]	Definition of EcucParamConfContainerDef EthSwtPortOutboundVlanPriorityAssignment
[ECUC_EthSwt_00181]	Definition of EcucStringParamDef EthSwtStreamFilterIEEE1722StreamId
[ECUC_EthSwt_00192]	Definition of EcucIntegerParamDef EthSwtPortOutboundVlanPriorityAssignmentRegeneratedPriority
[ECUC_EthSwt_00193]	Definition of EcucIntegerParamDef EthSwtPortOutboundVlanPriorityAssignmentOutboundVlanPriority
[SWS_EthSwt_00179]	
[SWS_EthSwt_00245]	
[SWS_EthSwt_00451]	
[SWS_EthSwt_00455]	
[SWS_EthSwt_00456]	
[SWS_EthSwt_00459]	

▽

△

Number	Heading
[SWS_EthSwt_00460]	
[SWS_EthSwt_00501]	

Table A.11: Deleted Specification Items in R24-11

A.3.4 Added Constraints in R24-11

Number	Heading
[SWS_EthSwt_CONSTR_00446]	SVL predefined MAC address configuration
[SWS_EthSwt_CONSTR_00447]	IVL predefined MAC address configuration
[SWS_EthSwt_CONSTR_00527]	Ethernet switch hardware support for frame preemption
[SWS_EthSwt_CONSTR_00528]	Frame preemption enabling per `EthSwtPort`
[SWS_EthSwt_CONSTR_00529]	Frame preemption status classification of preemptable Ethernet frames on egress per traffic class
[SWS_EthSwt_CONSTR_00530]	Frame preemption status classification of express Ethernet frames on egress per traffic class
[SWS_EthSwt_CONSTR_00532]	Value of priority to traffic class assignment should respect configured limitations
[SWS_EthSwt_CONSTR_00533]	Value of traffic class assignment should respect configured limitations
[SWS_EthSwt_CONSTR_00534]	Availability of a default traffic class per `EthSwtPortEgress`

▽

△

Number	Heading
[SWS_EthSwt_CONSTR_00536]	Traffic class to egress port queue assignment
[SWS_EthSwt_CONSTR_00538]	Definition of neighboring egress port queues
[SWS_EthSwt_CONSTR_00544]	Egress configuration constraint for scheduling with enhanced traffic shaping
[SWS_EthSwt_CONSTR_00545]	Enhanded traffic shaping require at least two egress port queues
[SWS_EthSwt_CONSTR_00546]	Neighboring egress port queues need the same variant of availability bandwidth configuration
[SWS_EthSwt_CONSTR_00552]	Constraint for configuration of available bandwidth in percent

Table A.12: Added Constraints in R24-11

A.3.5 Changed Constraints in R24-11

none

A.3.6 Deleted Constraints in R24-11

Number	Heading
[SWS_EthSwt_CONSTR_00413]	
[SWS_EthSwt_CONSTR_00457]	

▽

△

Number	Heading
[SWS_EthSwt_CONSTR_00495]	

Table A.13: Deleted Constraints in R24-11

A.4 Traceable item history of this document according to AUTOSAR Release R25-11

A.4.1 Added Specification Items in R25-11

Number	Heading
[ECUC_EthSwt_00260]	Definition of EcucBooleanParamDef EthSwtMacSecApi
[ECUC_EthSwt_00261]	Definition of EcucFloatParamDef EthSwtMainFunctionMacSecPeriod
[ECUC_EthSwt_00262]	Definition of EcucIntegerParamDef EthSwtPortMacSecBypassEtherType
[ECUC_EthSwt_00263]	Definition of EcucIntegerParamDef EthSwtPortMacSecBypassVlan
[ECUC_EthSwt_00264]	Definition of EcucBooleanParamDef EthSwtPortMacSecEnabled
[ECUC_EthSwt_00265]	Definition of EcucStringParamDef EthSwtPortMacSecBypassDestinationMacAddress
[SWS_EthSwt_00559]	Translating to default VLAN ID
[SWS_EthSwt_00564]	Initialize Ethernet switch ports with MACsec support, where MACsec functionality is enabled
[SWS_EthSwt_00565]	Initialize Ethernet switch ports with MACsec support, where MACsec functionality is disabled
[SWS_EthSwt_00566]	Handling if `EthSwt_MacSecInitRxSc` is called and `EthSwtPortMacSecEnabled` is available and set to `TRUE`
[SWS_EthSwt_00567]	Handling if `EthSwt_MacSecInitRxSc` is called and `EthSwtPortMacSecEnabled` is set to `FALSE` or the configuration parameter is not available
[SWS_EthSwt_00568]	Handling if `EthSwt_MacSecUpdateSecY` is called and `EthSwtPortMacSecEnabled` is available and set to `TRUE`
[SWS_EthSwt_00569]	Handling if attempts to update an Ethernet switch port with a MAC security entity configuration and the establishment of a transmit secure channel has been finalized

▽

△

Number	Heading
[SWS_EthSwt_00570]	Handling if EthSwt_MacSecUpdateSecY is called and update of a MAC security entity configuration and the establishment of a transmit secure channel was already successfully finalized
[SWS_EthSwt_00571]	Handling if EthSwt_MacSecUpdateSecY is called and EthSwtPortMacSecEnabled is set to FALSE or the configuration parameter is not available
[SWS_EthSwt_00572]	Handling if EthSwt_MacSecResetRxSc is called and EthSwtPortMacSecEnabled is available and set to TRUE
[SWS_EthSwt_00573]	Handling if EthSwt_MacSecResetRxSc is called and EthSwtPortMacSecEnabled is set to FALSE or the configuration parameter is not available
[SWS_EthSwt_00574]	Handling if EthSwt_MacSecAddRxSa is called and EthSwtPortMacSecEnabled is available and set to TRUE
[SWS_EthSwt_00575]	Handling if EthSwt_MacSecAddRxSa is called and EthSwtPortMacSecEnabled is set to FALSE or the configuration parameter is not available
[SWS_EthSwt_00576]	Handling if attempts to add a reception secure association at an Ethernet switch port and to enable or disable this secure association has been finalized
[SWS_EthSwt_00577]	Handling if EthSwt_MacSecUpdateRxSa is called with Active is set to TRUE, the secured association is deactivated and EthSwtPortMacSecEnabled is available and set to TRUE
[SWS_EthSwt_00578]	Handling if EthSwt_MacSecUpdateRxSa is called with Active is set to TRUE, the secured association is activated and EthSwtPortMacSecEnabled is available and set to TRUE
[SWS_EthSwt_00579]	Handling if EthSwt_MacSecUpdateRxSa is called with Active is set to FALSE, the secured association is activated and EthSwtPortMacSecEnabled is available and set to TRUE
[SWS_EthSwt_00580]	Handling if EthSwt_MacSecUpdateRxSa is called with Active is set to FALSE, the secured association is deactivated and EthSwtPortMacSecEnabled is available and set to TRUE
[SWS_EthSwt_00581]	Handling if EthSwt_MacSecUpdateRxSa is called and EthSwtPortMacSecEnabled is set to FALSE or the configuration parameter is not available
[SWS_EthSwt_00582]	Handling if EthSwt_MacSecDeleteRxSa is called and EthSwtPortMacSecEnabled is available and set to TRUE
[SWS_EthSwt_00583]	Handling if EthSwt_MacSecDeleteRxSa is called and EthSwtPortMacSecEnabled is set to FALSE or the configuration parameter is not available
[SWS_EthSwt_00584]	Handling if EthSwt_MacSecAddTxSa is called and EthSwtPortMacSecEnabled is available and set to TRUE
[SWS_EthSwt_00585]	Handling if EthSwt_MacSecAddTxSa is called and EthSwtPortMacSecEnabled is set to FALSE or the configuration parameter is not available
[SWS_EthSwt_00586]	Handling if attempts to enable / disable the transmission secure association at an Ethernet switch port has been finalized

▽

△

Number	Heading
[SWS_EthSwt_00587]	Handling if `EthSwt_MacSecUpdateTxSa` is called with `Active` is set to `TRUE`, the secured association is deactivated and `EthSwtPortMacSecEnabled` is available and set to `TRUE`
[SWS_EthSwt_00588]	Handling if `EthSwt_MacSecUpdateTxSa` is called with `Active` is set to `TRUE`, the secured association is activated and `EthSwtPortMacSecEnabled` is available and set to `TRUE`
[SWS_EthSwt_00589]	Handling if `EthSwt_MacSecUpdateTxSa` is called with `Active` is set to `FALSE`, the secured association is activated and `EthSwtPortMacSecEnabled` is available and set to `TRUE`
[SWS_EthSwt_00590]	Handling if `EthSwt_MacSecUpdateTxSa` is called with `Active` is set to `FALSE`, the secured association is deactivated and `EthSwtPortMacSecEnabled` is available and set to `TRUE`
[SWS_EthSwt_00591]	Handling if `EthSwt_MacSecUpdateTxSa` is called and `EthSwtPortMacSecEnabled` is set to `FALSE` or the configuration parameter is not available
[SWS_EthSwt_00592]	Handling if `EthSwt_MacSecDeleteTxSa` is called and `EthSwtPortMacSecEnabled` is available and set to `TRUE`
[SWS_EthSwt_00593]	Handling if `EthSwt_MacSecDeleteTxSa` is called and `EthSwtPortMacSecEnabled` is set to `FALSE` or the configuration parameter is not available
[SWS_EthSwt_00594]	Handling if `EthSwt_MacSecGetTxSaNextPn` is called and `EthSwtPortMacSecEnabled` is available and set to `TRUE`
[SWS_EthSwt_00595]	Handling if `EthSwt_MacSecGetTxSaNextPn` is called and `EthSwtPortMacSecEnabled` is set to `FALSE` or the configuration parameter is not available
[SWS_EthSwt_00596]	Handling if `EthSwt_MacSecGetMacSecStatistics` is called and `EthSwtPortMacSecEnabled` is available and set to `TRUE`
[SWS_EthSwt_00597]	Handling if `EthSwt_MacSecGetMacSecStatistics` is called and `EthSwtPortMacSecEnabled` is set to `FALSE` or the configuration parameter is not available
[SWS_EthSwt_00598]	Handling if attempts to collect MACsec statistics at an Ethernet switch port has been finalized
[SWS_EthSwt_00599]	Handling if `EthSwt_MacSecSetControlledPortEnabled` is called and `EthSwtPortMacSecEnabled` is available and set to `TRUE`
[SWS_EthSwt_00600]	Handling if `EthSwt_MacSecSetControlledPortEnabled` is called and `EthSwtPortMacSecEnabled` is set to `FALSE` or the configuration parameter is not available

Table A.14: Added Specification Items in R25-11

A.4.2 Changed Specification Items in R25-11

Number	Heading
[ECUC_EthSwt_00003]	Definition of EcucParamConfContainerDef EthSwtGeneral
[ECUC_EthSwt_00005]	Definition of EcucParamConfContainerDef EthSwtPort
[ECUC_EthSwt_00007]	Definition of EcucParamConfContainerDef EthSwtPortEgress
[ECUC_EthSwt_00014]	Definition of EcucParamConfContainerDef EthSwtPortIngress
[ECUC_EthSwt_00150]	Definition of EcucEnumerationParamDef EthSwtStreamFilterAction DestinationPortModificationType
[ECUC_EthSwt_00166]	Definition of EcucStringParamDef EthSwtStreamFilterMACAddressKey
[ECUC_EthSwt_00167]	Definition of EcucStringParamDef EthSwtStreamFilterMACAddressMask
[ECUC_EthSwt_00168]	Definition of EcucIntegerParamDef EthSwtStreamFilterVlanId
[ECUC_EthSwt_00169]	Definition of EcucIntegerParamDef EthSwtStreamFilterVlanPriority
[ECUC_EthSwt_00187]	Definition of EcucParamConfContainerDef EthSwtPortEgressQueue TransmissionSelectionCBSConfig
[ECUC_EthSwt_00199]	Definition of EcucParamConfContainerDef EthSwtVlanMembership
[ECUC_EthSwt_00206]	Definition of EcucStringParamDef EthSwtPredefinedMacAddress
[ECUC_EthSwt_00240]	Definition of EcucReferenceDef EthSwtDestPortsForUnknownUnicastMac DestAddressRef
[ECUC_EthSwt_00241]	Definition of EcucReferenceDef EthSwtDestPortsForUnknownMulticastMac DestAddressRef
[SWS_EthSwt_00098]	Definition of optional interfaces requested by module EthSwt
[SWS_EthSwt_00216]	Definition of API function EthSwt_ReadTrcvRegister
[SWS_EthSwt_00221]	Definition of API function EthSwt_WriteTrcvRegister
[SWS_EthSwt_00503]	
[SWS_EthSwt_00504]	
[SWS_EthSwt_00542]	Emission opportunity suspension with strict priority scheduling
[SWS_EthSwt_00549]	Tolerance of egress port queue emission within the defined measurement interval if available bandwidth is configured in percent is used
[SWS_EthSwt_00612]	
[SWS_EthSwt_91004]	Definition of API function EthSwt_EthRxProcessFrame
[SWS_EthSwt_91005]	Definition of API function EthSwt_EthRxFinishedIndication
[SWS_EthSwt_91006]	Definition of API function EthSwt_EthTxPrepareFrame

▽

△

Number	Heading
[SWS_EthSwt_91007]	Definition of API function EthSwt_EthTxAdaptBufferLength
[SWS_EthSwt_91009]	Definition of API function EthSwt_EthTxProcessFrame
[SWS_EthSwt_91010]	Definition of API function EthSwt_EthTxFinishedIndication
[SWS_EthSwt_91038]	Definition of API function EthSwt_GetRxMgmtObject
[SWS_EthSwt_91052]	Definition of API function EthSwt_ReadMmd
[SWS_EthSwt_91053]	Definition of API function EthSwt_WriteMmd

Table A.15: Changed Specification Items in R25-11

A.4.3 Deleted Specification Items in R25-11

Number	Heading
[ECUC_EthSwt_00009]	Definition of EcucReferenceDef EthSwtPortEgressPredecessorFifoRef
[ECUC_EthSwt_00011]	Definition of EcucParamConfContainerDef EthSwtPortFifo
[ECUC_EthSwt_00012]	Definition of EcucIntegerParamDef EthSwtPortFifoTrafficClassAssignment
[ECUC_EthSwt_00021]	Definition of EcucParamConfContainerDef EthSwtPortShaper
[ECUC_EthSwt_00042]	Definition of EcucIntegerParamDef EthSwtPortShaperIdleSlope
[ECUC_EthSwt_00074]	Definition of EcucParamConfContainerDef EthSwtPortPolicer
[ECUC_EthSwt_00075]	Definition of EcucIntegerParamDef EthSwtPortRatePolicedByteCount
[ECUC_EthSwt_00076]	Definition of EcucFloatParamDef EthSwtPortRatePolicedTimeInterval
[ECUC_EthSwt_00077]	Definition of EcucIntegerParamDef EthSwtPortRatePolicedPriority
[ECUC_EthSwt_00078]	Definition of EcucEnumerationParamDef EthSwtPortRateViolationAction
[ECUC_EthSwt_00081]	Definition of EcucReferenceDef EthSwtRateVlanMembershipRef
[ECUC_EthSwt_00098]	Definition of EcucIntegerParamDef EthSwtPortFifoMinimumLength
[ECUC_EthSwt_00132]	Definition of EcucIntegerParamDef EthSwtPortEgressFifoIdx
[ECUC_EthSwt_00189]	Definition of EcucIntegerParamDef EthSwtPortEgressQueueCreditBased ShaperUpperBoundary

▽

△

Number	Heading
[ECUC_EthSwt_00190]	Definition of EcucIntegerParamDef EthSwtPortEgressQueueCreditBasedShaperLowerBoundary
[SWS_EthSwt_00132]	
[SWS_EthSwt_00133]	
[SWS_EthSwt_00134]	
[SWS_EthSwt_00135]	
[SWS_EthSwt_00178]	
[SWS_EthSwt_91135]	Definition of callback function EthSwt_MacSecUpdateSecYNotification
[SWS_EthSwt_91136]	Definition of callback function EthSwt_MacSecAddTxSaNotification
[SWS_EthSwt_91137]	Definition of callback function EthSwt_MacSecAddRxSaNotification
[SWS_EthSwt_91138]	Definition of callback function EthSwt_MacSecGetMacSecStatisticsNotification

Table A.16: Deleted Specification Items in R25-11

A.4.4 Added Constraints in R25-11

none

A.4.5 Changed Constraints in R25-11

none

A.4.6 Deleted Constraints in R25-11

none

A.5 Traceable item history of this document according to AUTOSAR Release R26-11

A.5.1 Added Specification Items in R26-11

Number	Heading
[SWS_EthSwt_00529]	Frame preemption status classification of preemptable Ethernet frames on egress per traffic class

▽

△

Number	Heading
[SWS_EthSwt_00530]	Frame preemption status classification of express Ethernet frames on egress per traffic class

Table A.17: Added Specification Items in R26-11

A.5.2 Changed Specification Items in R26-11

Number	Heading
[ECUC_EthSwt_00006]	Definition of EcucReferenceDef ETHSWT_E_ACCESS
[ECUC_EthSwt_00125]	Definition of EcucReferenceDef ETHSWT_E_SYNCPORT2PHY
[SWS_EthSwt_00002]	Definition of imported datatypes of module EthSwt
[SWS_EthSwt_00113]	Ethernet Switch Access Failure
[SWS_EthSwt_00114]	Definition of scheduled function EthSwt_MainFunction
[SWS_EthSwt_00395]	Ethernet switch port and the referenced Ethernet transceiver are in contradicting modes.
[SWS_EthSwt_00555]	Translating VLAN IDs
[SWS_EthSwt_00572]	Handling if `EthSwt_MacSecResetRxSc` is called and `EthSwtPortMacSecEnabled` is available and set to `TRUE`
[SWS_EthSwt_91007]	Definition of API function EthSwt_EthTxAdaptBufferLength
[SWS_EthSwt_91104]	Definition of API function EthSwt_BackgroundTask

Table A.18: Changed Specification Items in R26-11

A.5.3 Deleted Specification Items in R26-11

Number	Heading
[ECUC_EthSwt_00031]	Definition of EcucBooleanParamDef EthSwtVersionInfoApi
[SWS_EthSwt_00058]	Definition of API function EthSwt_GetVersionInfo
[SWS_EthSwt_00124]	

Table A.19: Deleted Specification Items in R26-11

A.5.4 Added Constraints in R26-11

Number	Heading
[SWS_EthSwt_CONSTR_00615]	DefaultVlan must be configured if used
[SWS_EthSwt_CONSTR_00616]	Exclude host port and uplink ports from MACsec protection on a point to point connection

Table A.20: Added Constraints in R26-11

A.5.5 Changed Constraints in R26-11

none

A.5.6 Deleted Constraints in R26-11

Number	Heading
[SWS_EthSwt_CONSTR_00529]	Frame preemption status classification of preemptable Ethernet frames on egress per traffic class
[SWS_EthSwt_CONSTR_00530]	Frame preemption status classification of express Ethernet frames on egress per traffic class

Table A.21: Deleted Constraints in R26-11