

<b>Document Title</b>	Specification of Wireless Ethernet Driver
<b>Document Owner</b>	AUTOSAR
<b>Document Responsibility</b>	AUTOSAR
<b>Document Identification No</b>	798

<b>Document Status</b>	published
<b>Part of AUTOSAR Standard</b>	Classic Platform
<b>Part of Standard Release</b>	R23-11

<b>Document Change History</b>			
<b>Date</b>	<b>Release</b>	<b>Changed by</b>	<b>Description</b>
2023-11-23	R23-11	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>• Removal of Tx confirmation and transaction ID</li> <li>• WEthCtrlId no anymore manually configured</li> </ul>
2022-11-24	R22-11	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>• Figures are now at the begin of a chapter.</li> <li>• Fixed text alignment of some artifacts.</li> <li>• Fixed some incorrect trace references.</li> </ul>
2021-11-25	R21-11	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>• No content changes.</li> </ul>
2020-11-30	R20-11	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>• Relaxed requirements on base address and memory alignment.</li> <li>• References to the Ethernet Driver substituted by their content.</li> </ul>
2019-11-28	R19-11	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>• Operation for DCC Access queue modified.</li> <li>• Partition handling released.</li> <li>• Changed Document Status from Final to published.</li> </ul>
2018-10-31	4.4.0	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>• Basic Software Multicore Distribution (DRAFT).</li> </ul>



△

2017-12-08	4.3.1	AUTOSAR Release Management	<ul style="list-style-type: none"><li>• editorial changes</li></ul>
2016-11-30	4.3.0	AUTOSAR Release Management	<ul style="list-style-type: none"><li>• Initial Release.</li></ul>

## Disclaimer

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

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

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

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

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

## Contents

1	Introduction and functional overview	6
2	Acronyms and Abbreviations	8
3	Related documentation	9
3.1	Input documents & related standards and norms	9
3.2	Related specification	9
4	Constraints and assumptions	10
4.1	Limitations	10
4.2	Applicability to car domains	10
5	Dependencies to other modules	11
5.1	Driver Services	11
6	Requirements Tracing	12
7	Functional specification	14
7.1	Wireless Ethernet BSW stack	14
7.1.1	Indexing scheme	14
7.1.2	Transceiver configuration	14
7.1.3	General Requirements	15
7.1.4	Controller on-packet-base parameters	15
7.1.5	Key/Value Parameter Mapping	16
7.1.6	V2X Specific Controller Requirements	16
7.2	Error Classification	17
7.2.1	Development Errors	17
7.2.2	Runtime Errors	17
7.2.3	Transient Faults	18
7.2.4	Production Errors	18
7.2.5	Extended Production Errors	18
7.3	Security Events	18
8	API specification	19
8.1	Imported types	20
8.2	Type definitions	21
8.2.1	WEth_ConfigType	21
8.2.2	WEth_BufWRxParamIdType	22
8.2.3	WEth_BufWTxParamIdType	22
8.3	Function definitions	22
8.3.1	WEth_Init	23
8.3.2	WEth_SetControllerMode	24
8.3.3	WEth_GetControllerMode	25
8.3.4	WEth_GetPhysAddr	26
8.3.5	WEth_SetPhysAddr	26

8.3.6	WEth_UpdatePhysAddrFilter	27
8.3.7	WEth_ProvideTxBuffer	29
8.3.8	WEth_Transmit	30
8.3.9	WEth_TxConfirmation	31
8.3.10	WEth_Receive	32
8.3.11	WEth_GetWETHERStats32	33
8.3.12	WEth_GetWETHERStats64	35
8.3.13	WEth_WriteTrcvRegs	36
8.3.14	WEth_ReadTrcvRegs	37
8.3.15	WEth_GetBufWRxParams	38
8.3.16	WEth_GetBufWTxParams	39
8.3.17	WEth_SetBufWTxParams	40
8.3.18	WEth_GetVersionInfo	41
8.3.19	WEth_TriggerPriorityQueueTransmit	41
8.4	Callback notifications	42
8.5	Scheduled functions	42
8.5.1	WEth_MainFunction	42
8.6	Expected interfaces	42
8.6.1	Mandatory Interfaces	42
8.6.2	Optional Interfaces	43
8.6.3	Configurable interfaces	43
9	Sequence diagrams	44
10	Configuration specification	45
10.1	Containers and configuration parameters	45
10.1.1	Variants	45
10.2	WEth	45
10.3	WEthConfigSet	45
10.4	WEthCtrlConfig	46
10.5	WEthDemEventParameterRefs	49
10.6	WEthGeneral	51
A	Change history of AUTOSAR traceable items	55
A.1	Traceable item history of this document according to AUTOSAR Release R23-11	55
A.1.1	Added Specification Items in R23-11	55
A.1.2	Changed Specification Items in R23-11	55
A.1.3	Deleted Specification Items in R23-11	55
A.1.4	Added Constraints in R23-11	55
A.1.5	Changed Constraints in R23-11	55
A.1.6	Deleted Constraints in R23-11	55
A.2	Traceable item history of this document according to AUTOSAR Release R22-11	56
A.2.1	Added Constraints in R22-11	56
A.2.2	Changed Constraints in R22-11	56
A.2.3	Deleted Constraints in R22-11	56

# 1 Introduction and functional overview

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

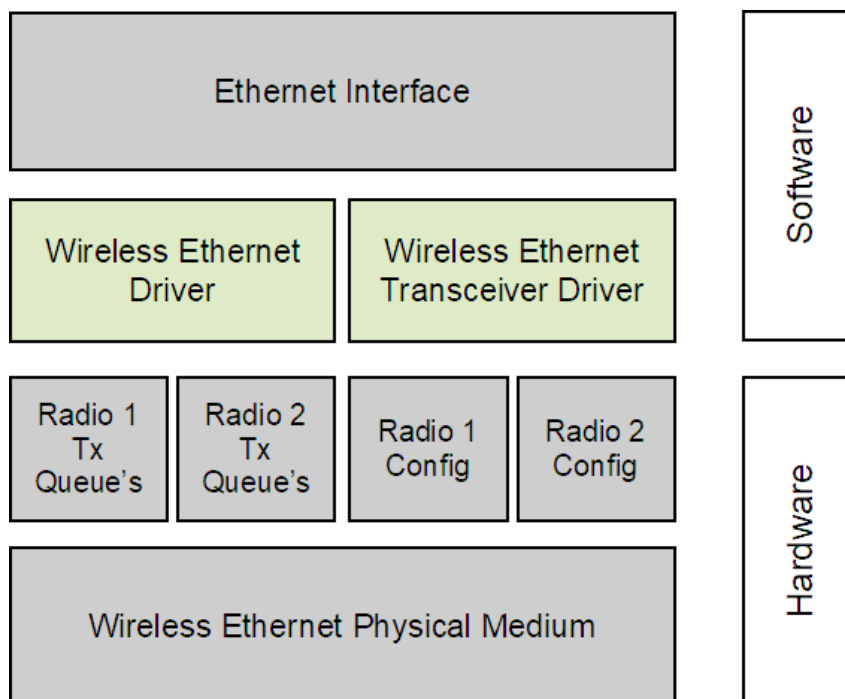
In the AUTOSAR Layered Software Architecture, the Wireless Ethernet driver belongs to the Microcontroller Abstraction Layer, or more precisely, to the Communication Drivers.

This indicates the main task of the Wireless Ethernet driver:

Provide to the upper layer (Ethernet Interface) a hardware independent interface comprising multiple equal controllers. This interface shall be uniform for all controllers. Thus, the upper layer (Ethernet Interface) may access the underlying bus system in a uniform manner. The interface provides functionality for initialization, configuration and data transmission. The configuration of the Wireless Ethernet Driver however is bus specific, since it takes into account the specific features of the communication controller.

A single Wireless Ethernet driver module supports only one type of controller hardware. The Wireless Ethernet driver's prefix requires a unique namespace. The Ethernet Interface can access different controller types using different Wireless Ethernet drivers using this prefix. The decision which driver to use to access a particular controller is a configuration parameter of the Ethernet Interface.

Figure 1.1 depicts the lower part of the Wireless Ethernet stack. One Ethernet Interface can access several radios using several Wireless Ethernet Transceiver drivers. Each radio may support multiple contexts i.e. multiple radio channel configurations.



**Figure 1.1: Wireless Ethernet stack module overview**

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

## 2 Acronyms and Abbreviations

The glossary below includes acronyms and abbreviations relevant to the WEth module that are not included in the AUTOSAR glossary [1].

Abbreviation / Acronym:	Description:
FCS	Frame Check Sequence
EthIf	Ethernet Interface (AUTOSAR BSW module)
Eth	Ethernet Driver (AUTOSAR BSW module)
ISR	Interrupt Service Routine
MCG	Module Configuration Generator
WEth	Wireless Ethernet Driver (AUTOSAR BSW module)
WEthTrcv	Wireless Ethernet Transceiver (AUTOSAR BSW module)

**Table 2.1: Acronyms and abbreviations used in the scope of this Document**



## 3 Related documentation

### 3.1 Input documents & related standards and norms

- [1] Glossary  
AUTOSAR\_FO\_TR\_Glossary
- [2] IEC:The Basic Model, IEC Norm
- [3] General Specification of Basic Software Modules  
AUTOSAR\_CP\_SWS\_BSWGeneral
- [4] Specification of ECU State Manager  
AUTOSAR\_CP\_SWS\_ECUSTateManager
- [5] Specification of Ethernet Driver  
AUTOSAR\_CP\_SWS\_EthernetDriver
- [6] Requirements on Vehicle-2-X Communication  
AUTOSAR\_CP\_SRS\_V2XCommunication
- [7] TS 102 724 V1.1.1: Intelligent Transport Systems (ITS); Harmonized Channel Specifications for Intelligent Transport Systems operating in the 5 GHz frequency band
- [8] Specification of Default Error Tracer  
AUTOSAR\_CP\_SWS\_DefaultErrorTracer
- [9] Specification of Ethernet Interface  
AUTOSAR\_CP\_SWS\_EthernetInterface
- [10] IEEE 802.11-2012

### 3.2 Related specification

AUTOSAR provides a General Specification on Basic Software modules [3, SWS BSW General], which is also valid for SWS\_WirelessEthernetDriver.

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

## 4 Constraints and assumptions

### 4.1 Limitations

- It is not possible to transmit data which exceeds the available buffer size of the used controller.
- AUTOSAR supports currently only wireless communication using IEEE 802.11p. Other 802.11 standards (e.g. for infrastructure networks and integration with TCP/IP) can be extended in future releases of the AUTOSAR standard.
- The V2X modules follow the guidance regarding the Day-1 scenarios defined by the Basic System Standards Profile from Car-2-Car-Communication-Consortium.
- AUTOSAR R20-11 only focuses on the European version of car-to-car communication as defined by ETSI. Extension to other regions are planned for future releases of the AUTOSAR standard.
- The Microcontroller Abstraction Layer Multi-Core Distribution Concept is implemented as "draft" in this software specification. Refer to chapter 10 for more information.

### 4.2 Applicability to car domains

The Wireless Ethernet Driver is intended to be used for wireless access of customer hardware (Access Point) and for wireless access of Vehicle-2-X (V2X) applications / BSW Modules (using a meshed network).

## 5 Dependencies to other modules

This chapter lists the modules interacting with the Wireless Ethernet Driver module.

Modules that use Wireless Ethernet Driver module:

- Ethernet Interface (EthIf)
- Wireless Ethernet Transceiver (WEthTrcv)

Modules used by the Wireless Ethernet Driver module:

- Typically, the wireless radio hardware is an external device that is accessed by an existing communication driver such as SPI.

### 5.1 Driver Services

**[SWS\_WEth\_10001]** [If the Wireless Ethernet controller is on-chip, the Wireless Ethernet Driver module shall not use any service of other drivers.]()

**[SWS\_WEth\_10003]** [If an off-chip Wireless Ethernet controller is used<sup>1</sup>, the Wireless Ethernet driver shall use services of other MCAL drivers (e.g. SPI).]()

Implementation hint: If the Wireless Ethernet driver uses services of other MCAL drivers (e.g. SPI), it must be ensured that these drivers are up and running before initializing the Wireless Ethernet driver. The sequence of initialization of different drivers is partly specified in the [4, ECU State Manager].

**[SWS\_WEth\_10004]** [All the Wireless Ethernet driver interfaces shall be implemented in a non-blocking manner. In cases where the action can be performed immediately and atomically, the confirmation is reported in the request function's return code. Alternatively, the initiation of an action is performed by a call to a 'request' function and the result of the action is reported by a corresponding 'confirm' callback.]()

---

<sup>1</sup>In this case the Wireless Ethernet driver is not any more part of the  $\mu$ C abstraction layer but put part of the ECU abstraction layer. Therefore it is (theoretically) allowed to use any  $\mu$ C abstraction layer driver it needs

## 6 Requirements Tracing

Note:

Requirement IDs within this document have an encoding to state where each requirement has its origin:

- SWS items starting with a leading 0 (SWS\_WEth\_0xxxx) are inherited from the [5, SWS Ethernet Driver].
- SWS items starting with a leading 1 (SWS\_WEth\_1xxxx) are module specific and not inherited.
- SWS items starting with a leading 2 (SWS\_WEth\_2xxxx) are inherited from C2C-CC Basic System Profile

Requirement	Description	Satisfied by
[SRS_BSW_00323]	All AUTOSAR Basic Software Modules shall check passed API parameters for validity	[SWS_WEth_00008]
[SRS_BSW_00327]	Error values naming convention	[SWS_WEth_00016]
[SRS_BSW_00333]	For each callback function it shall be specified if it is called from interrupt context or not	[SWS_WEth_00244]
[SRS_BSW_00339]	Reporting of production relevant error status	[SWS_WEth_00173]
[SRS_BSW_00359]	All AUTOSAR Basic Software Modules callback functions shall avoid return types other than void if possible	[SWS_WEth_00243]
[SRS_BSW_00413]	An index-based accessing of the instances of BSW modules shall be done	[SWS_WEth_00003]
[SRS_BSW_00487]	Errors for module initialization shall follow a naming rule	[SWS_WEth_10039] [SWS_WEth_10046]
[SRS_V2X_00010]	The implementation of the V2X system shall follow additional guidance given by C2C-CC requirements	[SWS_WEth_20235]
[SRS_V2X_00176]	The V2X system shall change pseudonyms	[SWS_WEth_10073]
[SRS_V2X_00214]	The V2X system shall allow applications to deactivate transmission of CAMs	[SWS_WEth_00004]
[SRS_V2X_00242]	The V2Xsystem shall manage CAM transmission in such a way, that no outdated CAM will be transmitted	[SWS_WEth_20242]
[SRS_V2X_00245]	The V2X system shall support per-packet transmission power control	[SWS_WEth_10013] [SWS_WEth_10037] [SWS_WEth_10051]





Requirement	Description	Satisfied by
[SRS_V2X_00391]	The V2X system's access layer shall be ITS-G5 compliant	[SWS_WEth_10005] [SWS_WEth_10006] [SWS_WEth_10009] [SWS_WEth_10038] [SWS_WEth_10045] [SWS_WEth_10052] [SWS_WEth_10064] [SWS_WEth_10065] [SWS_WEth_10066] [SWS_WEth_10067] [SWS_WEth_10068]
[SRS_V2X_00451]	The V2X system's access layer shall be compliant to the ETSI Harmonized Channel Specifications	[SWS_WEth_10007] [SWS_WEth_10008] [SWS_WEth_10069]

**Table 6.1: RequirementsTracing**

## 7 Functional specification

The Wireless Ethernet driver provides communications access to the radio for wireless communications. On transmission the driver writes the packet into an appropriate buffer inside the Wireless Ethernet driver, on packet reception the Wireless Ethernet driver calls the receive packet callback function with the packet contents as a parameter.

### 7.1 Wireless Ethernet BSW stack

As part of the AUTOSAR Layered Software Architecture (see Figure 1.1), the Wireless Ethernet BSW modules also form a layered software stack. The Ethernet Interface (EthIf) module accesses several controllers using the Wireless Ethernet Driver layer, which can be made up of several Wireless Ethernet Driver modules.

The Wireless Ethernet Driver supports Multi Core distribution for improved performance.

#### 7.1.1 Indexing scheme

Users of the Wireless Ethernet Driver identify controller resources using an indexing scheme as described in the [5, Ethernet Driver].

**[SWS\_WEth\_00003]** [The Wireless Ethernet Driver is using a zero-based index to abstract the access for upper software layers. The parameter WEth\_CtrlId within configuration corresponds to parameter CtrlId used in the API.] ([SRS\\_BSW\\_00413](#))

**[SWS\_WEth\_00004]** [A buffer index (BufId) identifies a Wireless Ethernet buffer processed by Wireless Ethernet Driver API functions. Each controller's buffers are identified by buffer indexes 0 to (n-1) where n is the number of buffers processed by the corresponding controller. Buffer indexes are valid within a tuple <CtrlId, BufId> only. A BufId uniquely identifies the buffer used for a Wireless Ethernet Driver.] ([SRS\\_V2X\\_00214](#))

#### 7.1.2 Transceiver configuration

**[SWS\_WEth\_10007]** [The Wireless Ethernet Driver shall provide an API that enables the Wireless Ethernet Transceiver to set the general radio specific parameters via an API WEth\_WriteTrcvRegs to the transceiver.] ([SRS\\_V2X\\_00451](#))

**[SWS\_WEth\_10008]** [The Wireless Ethernet Driver shall provide an API that enables the Wireless Ethernet Transceiver to get the general radio specific parameters via an API WEth\_ReadTrcvRegs from the transceiver.] ([SRS\\_V2X\\_00451](#))

### 7.1.3 General Requirements

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

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

**[SWS\_WEth\_10009]** [For reception the Wireless Ethernet Controller shall enable hardware capabilities to discard frames with incorrect Frame Check Sequence (FCS).] ([SRS\\_V2X\\_00391](#))

**[SWS\_WEth\_00243]** [Wireless Ethernet Driver shall call EthIf\_TxConfirmation to indicate a successful transmission from the Interrupt routine (if the notification has been enabled).] ([SRS\\_BSW\\_00359](#))

**[SWS\_WEth\_00244]** [Wireless Ethernet Driver shall call EthIf\_RxIndication to indicate a successful reception from the Interrupt routine.] ([SRS\\_BSW\\_00333](#))

### 7.1.4 Controller on-packet-base parameters

For the Wireless Ethernet Driver it is important to be able to configure the transmission and the reception parameters for a destined radio of the Wireless Ethernet Transceiver. This is not only needed as general configuration for the radio (e.g. for access points), it is also necessary to be able to configure the parameters on a per-packet-base (e.g. for 802.11p meshed networks).

**[SWS\_WEth\_10005]** [The Wireless Ethernet Driver shall provide an API WEth\_GetBufWRxParams that can provide a list of buffer based reception parameters.] ([SRS\\_V2X\\_00391](#))

**[SWS\_WEth\_10038]** [The API WEth\_GetBufWRxParams shall read properties of type WEth\_BufWRxParamIdType of the access layer properties of a received packet.] ([SRS\\_V2X\\_00391](#))

**[SWS\_WEth\_10037]** [The Wireless Ethernet Driver shall provide an API WEth\_GetBufWTxParams that can provide a list of buffer based transmission parameters.] ([SRS\\_V2X\\_00245](#))

**[SWS\_WEth\_10045]** [The API WEth\_GetBufWTxParams shall read properties of type WEth\_BufWTxParamIdType of the access layer properties of a received packet.] ([SRS\\_V2X\\_00391](#))

**[SWS\_WEth\_10006]** [The Wireless Ethernet Driver shall provide an API WEth\_SetBufWTxParams that sets a list of buffer based transmission parameters.] ([SRS\\_V2X\\_00391](#))

**[SWS\_WEth\_10052]** [The API WEth\_SetBufWTxParams shall set properties of type WEth\_BufWTxParamIdType of the access layer properties for a packet to be sent.] ([SRS\\_V2X\\_00391](#))

### 7.1.5 Key/Value Parameter Mapping

**[SWS\_WEth\_10064]** [For unique reference to transmission and reception parameters of a sent or received WEth packet, unique enumeration IDs shall be used within this module.] ([SRS\\_V2X\\_00391](#))

**[SWS\_WEth\_10065]** [Functions using the type WEth\_BufWRxParamIdType shall use a list of uint32 values for the list of corresponding values.] ([SRS\\_V2X\\_00391](#))

**[SWS\_WEth\_10066]** [

ParamId	ParamValue Type
WETH_BUFWRXPID_RSSI	uint8
WETH_BUFWRXPID_CHANNEL_ID	uint16
WETH_BUFWRXPID_FREQ	uint16
WETH_BUFWRXPID_ANTENNA_ID	uint8

Functions using the type WEth\_BufWRxParamIdType shall use the corresponding values of the table above for the type mapping.

] ([SRS\\_V2X\\_00391](#))

**[SWS\_WEth\_10067]** [Functions using the type WEth\_BufWTxParamIdType shall use a list of uint32 values for the list of corresponding values.] ([SRS\\_V2X\\_00391](#))

**[SWS\_WEth\_10068]** [

ParamId	ParamValue Type
WETH_BUFWTXPID_POWER	uint8
WETH_BUFWTXPID_CHANNEL_ID	uint16
WETH_BUFWTXPID_QUEUE_ID	uint8
WETH_BUFWTXPID_ANTENNA_ID	uint8

Functions using the WEth\_BufWTxParamIdType shall use the corresponding values of the table above for the type mapping.

] ([SRS\\_V2X\\_00391](#))

### 7.1.6 V2X Specific Controller Requirements

**[SWS\_WEth\_10069]** [The following requirements are only valid for WEth Controllers used within the [6, V2X Communication Stack].] ([SRS\\_V2X\\_00451](#))

**[SWS\_WEth\_20235]** [The WEth module shall support at least the following DCC-Profiles defined inside [7]: DP0, DP1, DP2 and DP3.

- DP0, used for TC = 0
- DP1: used for TC = 1



- DP2: used for TC = 2
- DP3: used for other low priority messages with TC > 2

]([SRS\\_V2X\\_00010](#))

**[SWS\_WEth\_20242]** [The WEth module shall discard a message with the DCC-Profile ID DP2 in the DCC\_Access queues if a new message with the DCC-Profile ID DP2 arrives in the DCC\_Access queues.] ([SRS\\_V2X\\_00242](#))

**[SWS\_WEth\_10073]** [The Wireless Ethernet Driver shall flush the transmit queues during a pseudonym change (call of WEth\_SetPhysAddr), to avoid transmitting packets with an old pseudonym.] ([SRS\\_V2X\\_00176](#))

## 7.2 Error Classification

Section "Error Handling" of the document "General Specification of Basic Software Modules" [3] describes the error handling of the Basic Software in detail. Above all, it constitutes a classification scheme consisting of five error types which may occur in BSW modules.

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

**[SWS\_WEth\_00008]** [In case development error detection is enabled for the Wireless Ethernet Driver module: The Wireless Ethernet Driver module WEth shall check API parameters for validity and report detected errors to the DET [8].] ([SRS\\_BSW\\_00323](#))

### 7.2.1 Development Errors

In case development error detection is enabled for the Wireless Ethernet Driver module: The Wireless Ethernet Driver module shall check API parameters for validity and report detected errors to the DET.

**[SWS\_WEth\_00016] Definiton of development errors in module WEth [**

Type of error	Related error code	Error value
Invalid controller index	WETH_E_INV_CTRL_ID	0x01
WEth module was not initialized	WETH_E_UNINIT	0x02
Invalid pointer in parameter list	WETH_E_PARAM_POINTER	0x03

]([SRS\\_BSW\\_00327](#))

### 7.2.2 Runtime Errors

There are no runtime errors.

### 7.2.3 Transient Faults

There are no transient faults.

### 7.2.4 Production Errors

There are no production errors.

### 7.2.5 Extended Production Errors

Extended production errors are handled as events of the Diagnostic Event Manager. The event IDs are defined in the following tables, while the actual values are assigned externally by the configuration of the Diagnostic Event Manager, and are included in the module via Dem.h.

#### [SWS\_WEth\_00173] [

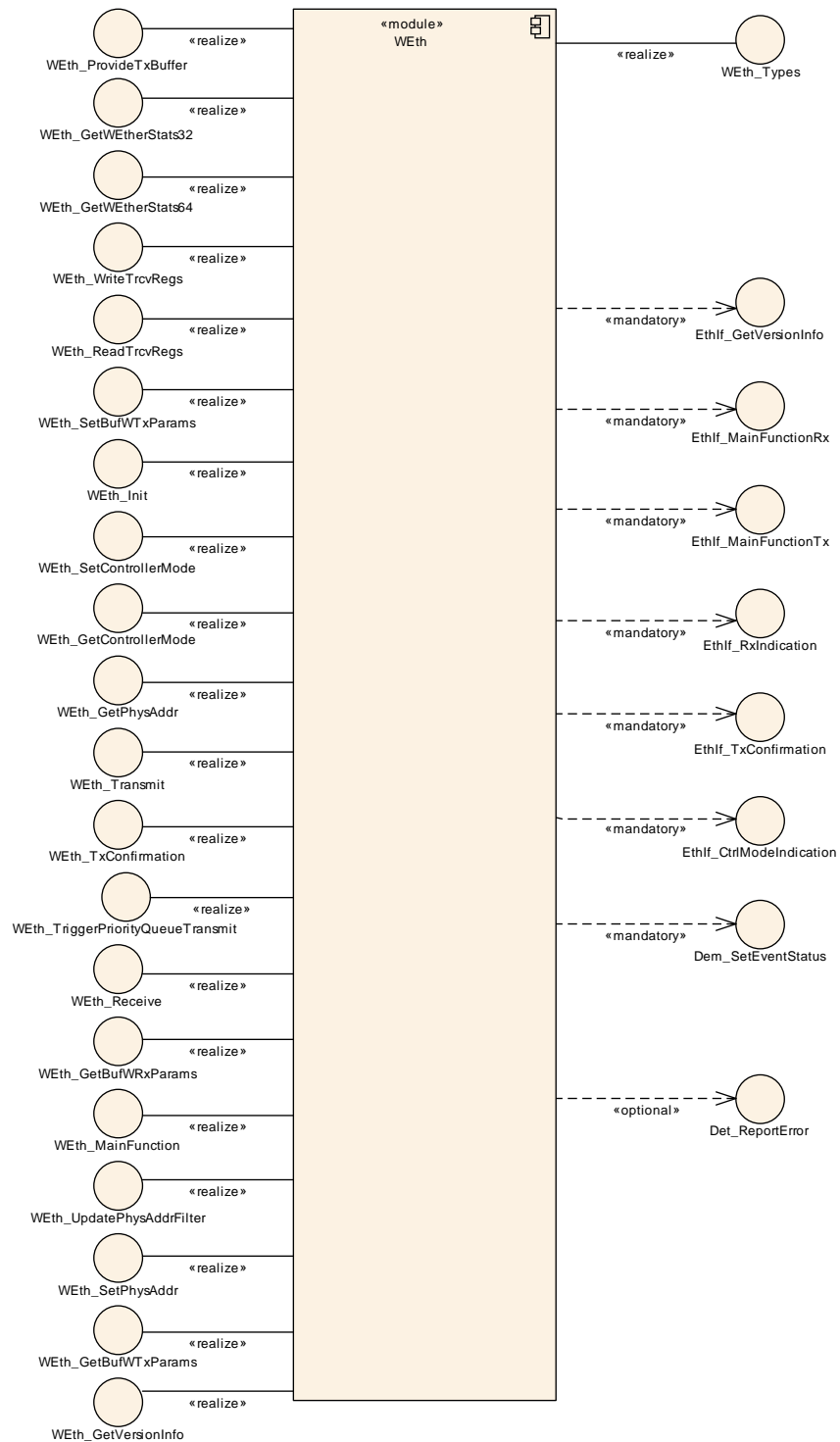
Error Name:	WETH_E_ACCESS	
Short Description:	Wireless Ethernet Controller Access Failure.	
Long Description:	Monitors the access to the Wireless Ethernet Controller in the context of the WEth_MainFunction..	
Detection Criteria:	Fail	When polling for state changes of the Wireless Ethernet Controller fails the module shall report the extended production error with event status DEM_EVENT_STATUS_PREFAILED to DEM.
	Pass	When polling for state changes of the Wireless Ethernet Controller succeeds the module shall report the extended production error with event status DEM_EVENT_STATUS_PREPASSED to DEM.
Secondary Parameters:	None.	
Time Required:	None.	
Monitor Frequency	None.	

](SRS\_BSW\_00339)

## 7.3 Security Events

The module does not report security events.

## 8 API specification



**Figure 8.1: Module dependencies of the WEth module**

## 8.1 Imported types

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

### [SWS\_WEth\_00026] Definition of imported datatypes of module WEth [

<i>Module</i>	<i>Header File</i>	<i>Imported Type</i>
ComStack_Types	ComStack_Types.h	BufReq_ReturnType
	ComStackTypes.h	TimeStampQualType (draft)
	ComStackTypes.h	TimeStampType (draft)
	ComStackTypes.h	TimeTupleType (draft)
Dem	Rte_Dem_Type.h	Dem_EventIdType
	Rte_Dem_Type.h	Dem_EventStatusType
Eth	Eth_GeneralTypes.h	Eth_BufIdxType
	Eth_GeneralTypes.h	Eth_DataType
	Eth_GeneralTypes.h	Eth_FilterActionType
	Eth_GeneralTypes.h	Eth_FrameType
	Eth_GeneralTypes.h	Eth_ModeType
	Eth_GeneralTypes.h	Eth_RxStatusType
Std	Std_Types.h	Std_ReturnType
	Std_Types.h	Std_VersionInfoType

]0

## 8.2 Type definitions



Figure 8.2: Shared typedefs of the WEth component

### 8.2.1 WEth\_ConfigType

[SWS\_WEth\_10011] Definition of datatype WEth\_ConfigType [

<b>Name</b>	WEth_ConfigType
<b>Kind</b>	Structure
<b>Description</b>	Implementation specific structure of the post build configuration
<b>Available via</b>	WEth.h

]()

### 8.2.2 WEth\_BufWRxParamIdType

[SWS\_WEth\_10012] Definition of datatype WEth\_BufWRxParamIdType [

<b>Name</b>	WEth_BufWRxParamIdType		
<b>Kind</b>	Enumeration		
<b>Range</b>	WETH_BUFWRXPID_RSSI	0x00	Parameter Id for RSSI value
	WETH_BUFWRXPID_CHANNEL_ID	0x01	Parameter Id for Channel Id. Channel Id values are specified within IEEE 802.11-2012 Annex E.
	WETH_BUFWRXPID_FREQ	0x02	Frequency on the channel with that the packet has been received
	WETH_BUFWRXPID_ANTENNA_ID	0x04	Index of the used antenna
<b>Description</b>	Wireless radio parameters for a packet that has been received.		
<b>Available via</b>	WEth_GeneralTypes.h		

]()

### 8.2.3 WEth\_BufWTxParamIdType

[SWS\_WEth\_10013] Definition of datatype WEth\_BufWTxParamIdType [

<b>Name</b>	WEth_BufWTxParamIdType		
<b>Kind</b>	Enumeration		
<b>Range</b>	WETH_BUFWTXPID_POWER	0x00	Parameter Id for transmit power
	WETH_BUFWTXPID_CHANNEL_ID	0x01	Parameter Id for Channel Id. Channel Id values are specified within IEEE 802.11-2012 Annex E.
	WETH_BUFWTXPID_QUEUE_ID	0x02	Queue index for ECDA / DCC queues
	WETH_BUFWTXPID_ANTENNA_ID	0x04	Index of the used antenna
<b>Description</b>	Wireless radio parameters for a packet that has to be transmitted.		
<b>Available via</b>	WEth_GeneralTypes.h		

]([SRS\\_V2X\\_00245](#))

## 8.3 Function definitions

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

### 8.3.1 WEth\_Init

#### [SWS\_WEth\_00027] Definition of API function WEth\_Init [

<b>Service Name</b>	WEth_Init	
<b>Syntax</b>	<pre>void WEth_Init (     const WEth_ConfigType* CfgPtr )</pre>	
<b>Service ID [hex]</b>	0x01	
<b>Sync/Async</b>	Synchronous	
<b>Reentrancy</b>	Non Reentrant	
<b>Parameters (in)</b>	CfgPtr	Points to the implementation specific structure
<b>Parameters (inout)</b>	None	
<b>Parameters (out)</b>	None	
<b>Return value</b>	None	
<b>Description</b>	Initializes the Wireless Ethernet Driver	
<b>Available via</b>	WEth.h	

]()

**[SWS\_WEth\_00028]** [The function shall store the access to the configuration structure for subsequent API calls.]()

**[SWS\_WEth\_00034]** [The function shall for all configured Wireless Ethernet controllers in the current WEthConfigSet:

- Disable all controller
- Clear pending Wireless Ethernet interrupts
- Configure all controller configuration parameters (e.g. interrupts, frame length, frame filter, ...)
- Configure all transmit / receive resources (e.g. buffer initialization)
- delete all pending transmit and receive requests

]()

**[SWS\_WEth\_00029]** [The function shall change the state of the component from WETH\_STATE\_UNINIT to WETH\_STATE\_INIT.]()

**[SWS\_WEth\_00039]** [The function shall check the access to the Wireless Ethernet controller. If the check fails, the function shall raise the production error WETH\_E\_ACCESS.]()

**[SWS\_WEth\_00031]** [Caveat: The API has to be called during initialization.]()

**[SWS\_WEth\_10002]** [The function WEth\_Init shall initialize all on-chip hardware resources that are used by the Wireless Ethernet controller.]()

### 8.3.2 WEth\_SetControllerMode

#### [SWS\_WEth\_00041] Definition of API function WEth\_SetControllerMode [

<b>Service Name</b>	WEth_SetControllerMode	
<b>Syntax</b>	Std_ReturnType WEth_SetControllerMode ( uint8 CtrlId, Eth_ModeType CtrlMode )	
<b>Service ID [hex]</b>	0x03	
<b>Sync/Async</b>	Asynchronous	
<b>Reentrancy</b>	Non Reentrant	
<b>Parameters (in)</b>	CtrlId	Index of the controller within the context of the Wireless Ethernet Driver
	CtrlMode	ETH_MODE_DOWN: disable the controller ETH_MODE_ACTIVE: enable the controller
<b>Parameters (inout)</b>	None	
<b>Parameters (out)</b>	None	
<b>Return value</b>	Std_ReturnType	E_OK: success E_NOT_OK: controller mode could not be changed
<b>Description</b>	Enables / disables the indexed controller	
<b>Available via</b>	WEth.h	

]()

#### [SWS\_WEth\_00042] [The function shall:

- Put the controller in the specified mode given in the parameter 'CtrlMode'
  - Upon mode ETH\_MODE\_DOWN the driver shall:
    - \* Disable the Wireless Ethernet controller
    - \* Reset all transmit and receive buffers (i.e. ignore all pending transmission and reception requests)
  - Upon mode ETH\_MODE\_ACTIVE:
    - \* Enable all transmit and receive buffers
    - \* Enable the Wireless Ethernet controller

]()

[SWS\_WEth\_00043] [If development error detection is enabled: the function shall check that the service WEth\_Init was previously called. If the check fails, the function shall raise the development error WETH\_E\_UNINIT.]()

[SWS\_WEth\_00044] [If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error WETH\_E\_INV\_CTRL\_IDX.]()

[SWS\_WEth\_00168] [The function shall check the access to the Wireless Ethernet controller. If the check fails, the function shall raise the production error WETH\_E\_ACCESS and return E\_NOT\_OK.]()



[SWS\_WEth\_00045] [Caveat: The function requires previous controller initialization (WEth\_Init).]()

### 8.3.3 WEth\_GetControllerMode

[SWS\_WEth\_00046] Definition of API function WEth\_GetControllerMode [

<b>Service Name</b>	WEth_GetControllerMode	
<b>Syntax</b>	Std_ReturnType WEth_GetControllerMode ( uint8 CtrlId, Eth_ModeType* CtrlModePtr )	
<b>Service ID [hex]</b>	0x04	
<b>Sync/Async</b>	Synchronous	
<b>Reentrancy</b>	Non Reentrant	
<b>Parameters (in)</b>	CtrlId	Index of the controller within the context of the Wireless Ethernet Driver
<b>Parameters (inout)</b>	None	
<b>Parameters (out)</b>	CtrlModePtr	ETH_MODE_DOWN: the controller is disabled ETH_MODE_ACTIVE: the controller is enabled
<b>Return value</b>	Std_ReturnType	E_OK: success E_NOT_OK: controller mode could not be obtained
<b>Description</b>	Obtains the state of the indexed controller	
<b>Available via</b>	WEth.h	

]()

[SWS\_WEth\_00047] [The function shall read the current controller mode.]()

[SWS\_WEth\_00048] [If development error detection is enabled: the function shall check that the service WEth\_Init was previously called. If the check fails, the function shall raise the development error WETH\_E\_UNINIT.]()

[SWS\_WEth\_00049] [If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error WETH\_E\_INV\_CTRL\_IDX.]()

[SWS\_WEth\_00050] [If development error detection is enabled: the function shall check the parameter CtrlModePtr for being valid. If the check fails, the function shall raise the development error WETH\_E\_PARAM\_POINTER.]()

[SWS\_WEth\_00051] [Caveat: The function requires previous controller initialization (WEth\_Init).]()

### 8.3.4 WEth\_GetPhysAddr

#### [SWS\_WEth\_00052] Definition of API function WEth\_GetPhysAddr [

<b>Service Name</b>	WEth_GetPhysAddr	
<b>Syntax</b>	<pre>void WEth_GetPhysAddr (     uint8 CtrlId,     uint8* PhysAddrPtr )</pre>	
<b>Service ID [hex]</b>	0x08	
<b>Sync/Async</b>	Synchronous	
<b>Reentrancy</b>	Non Reentrant	
<b>Parameters (in)</b>	CtrlId	Index of the controller within the context of the Wireless Ethernet Driver
<b>Parameters (inout)</b>	None	
<b>Parameters (out)</b>	PhysAddrPtr	Physical source address (MAC address) in network byte order.
<b>Return value</b>	void	None
<b>Description</b>	Obtains the physical source address used by the indexed controller	
<b>Available via</b>	WEth.h	

]()

**[SWS\_WEth\_00053]** [The function shall read the source address used by the indexed controller.]()

**[SWS\_WEth\_00054]** [If development error detection is enabled: the function shall check that the service WEth\_Init was previously called. If the check fails, the function shall raise the development error WETH\_E\_UNINIT.]()

**[SWS\_WEth\_00055]** [If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error WETH\_E\_INV\_CTRL\_IDX.]()

**[SWS\_WEth\_00056]** [If development error detection is enabled: the function shall check the parameter PhysAddrPtr for being valid. If the check fails, the function shall raise the development error WETH\_E\_PARAM\_POINTER.]()

**[SWS\_WEth\_00057]** [Caveat: The function requires previous controller initialization (WEth\_Init).]()

### 8.3.5 WEth\_SetPhysAddr

#### [SWS\_WEth\_00151] Definition of API function WEth\_SetPhysAddr [

<b>Service Name</b>	WEth_SetPhysAddr	
<b>Syntax</b>	<pre>void WEth_SetPhysAddr (     uint8 CtrlId,     const uint8* PhysAddrPtr )</pre>	





<b>Service ID [hex]</b>	0x13	
<b>Sync/Async</b>	Synchronous	
<b>Reentrancy</b>	Non Reentrant for the same CtrlId, reentrant for different	
<b>Parameters (in)</b>	CtrlId	Index of the controller within the context of the Wireless Ethernet Driver
	PhysAddrPtr	Pointer to memory containing the physical source address (MAC address) in network byte order.
<b>Parameters (inout)</b>	None	
<b>Parameters (out)</b>	None	
<b>Return value</b>	None	
<b>Description</b>	Sets the physical source address used by the indexed controller	
<b>Available via</b>	WEth.h	

]()

**[SWS\_WEth\_00139]** [The function shall update the source address used by the indexed controller.]()

**[SWS\_WEth\_00140]** [If development error detection is enabled: the function shall check that the service WEth\_Init was previously called. If the check fails, the function shall raise the development error WETH\_E\_UNINIT.]()

**[SWS\_WEth\_00141]** [If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error WETH\_E\_INV\_CTRL\_IDX.]()

**[SWS\_WEth\_00142]** [If development error detection is enabled: the function shall check the parameter PhysAddrPtr for being valid. If the check fails, the function shall raise the development error WETH\_E\_PARAM\_POINTER.]()

**[SWS\_WEth\_00143]** [Caveat: The function requires previous controller initialization (WEth\_Init).]()

### 8.3.6 WEth\_UpdatePhysAddrFilter

**[SWS\_WEth\_00152]** Definition of API function WEth\_UpdatePhysAddrFilter [

<b>Service Name</b>	WEth_UpdatePhysAddrFilter	
<b>Syntax</b>	<pre>Std_ReturnType WEth_UpdatePhysAddrFilter (     uint8 CtrlId,     const uint8* PhysAddrPtr,     const uint8* PhysMaskPtr,     Eth_FilterActionType Action )</pre>	
<b>Service ID [hex]</b>	0x12	
<b>Sync/Async</b>	Synchronous	
<b>Reentrancy</b>	Non Reentrant for the same CtrlId, reentrant for different	
<b>Parameters (in)</b>	CtrlId	Index of the context within the Wireless Ethernet Driver





	PhysAddrPtr	Pointer to memory containing the physical destination address (MAC address) in network byte order. This is the multicast destination address of the layer 2 Ethernet packet.
	PhysMaskPtr	Pointer to memory containing the mask value in network byte order.
	Action	Add or remove the address from the Wireless Ethernet controllers filter.
<b>Parameters (inout)</b>	None	
<b>Parameters (out)</b>	None	
<b>Return value</b>	Std_ReturnType	E_OK: filter was successfully changed E_NOT_OK: filter could not be changed
<b>Description</b>	Update the physical source address to/from the indexed context filter. If the Wireless Ethernet Controller is not capable to do the filtering, the software has to do this.	
<b>Available via</b>	WEth.h	

]()

**[SWS\_WEth\_00150]** [The function shall update the physical address receive filter of the indexed controller.]()

**[SWS\_WEth\_00245]** [The Wireless Ethernet driver module will receive a frame when the destination Address match the PhyAddrPtr passed here. (e.g matching can be done via hash table or simple pattern matching)]()

Note: Underlying HW mechanism can be used if available. Otherwise the Ethernet driver needs to do this by software.

**[SWS\_WEth\_00246]** [If the matching is positive, the upper layer shall be notified by calling RxIndication() callback.

If the matching is negative, the frame shall be discarded.]()

**[SWS\_WEth\_00164]** [If development error detection is enabled: the function shall check that the service WEth\_Init was previously called. If the check fails, the function shall raise the development error WETH\_E\_UNINIT.]()

**[SWS\_WEth\_00165]** [If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error WETH\_E\_INV\_CTRL\_IDX.]()

**[SWS\_WEth\_00166]** [If development error detection is enabled the function shall check the parameter PhysAddrPtr for being valid. If the check fails, the function shall raise the development error WETH\_E\_PARAM\_POINTER.]()

**[SWS\_WEth\_00167]** [Caveat: The function requires previous controller initialization (Eth\_Init).]()

**[SWS\_WEth\_00144]** [If the physical source address (MAC address) is set to FF:FF:FF:FF:FF:FF, this shall completely open the filter.]()

**[SWS\_WEth\_00146]** [If this API is used and the hardware does not support filtering, promiscuous mode shall be enabled during initialization.]()

**[SWS\_WEth\_00147]** [If the physical source address (MAC address) is set to 00:00:00:00:00:00, this shall reduce the filter to the controllers unique unicast MAC address and end promiscuous mode if it was turned on.]()

### 8.3.7 WEth\_ProvideTxBuffer

**[SWS\_WEth\_00077]** Definition of API function WEth\_ProvideTxBuffer [

<b>Service Name</b>	WEth_ProvideTxBuffer	
<b>Syntax</b>	<pre>BufReq_ReturnType WEth_ProvideTxBuffer (     uint8 CtrlId,     uint8 Priority,     Eth_BufIdxType* BufIdxPtr,     uint8** BufPtr,     uint16* LenBytePtr )</pre>	
<b>Service ID [hex]</b>	0x09	
<b>Sync/Async</b>	Synchronous	
<b>Reentrancy</b>	Non Reentrant	
<b>Parameters (in)</b>	CtrlId	Index of the context within the Wireless Ethernet Driver
	Priority	Priority value used for selection of different wireless transmit queues
<b>Parameters (inout)</b>	LenBytePtr	In: desired length in bytes, out: granted length in bytes
<b>Parameters (out)</b>	BufIdxPtr	Index to the granted buffer resource. To be used for subsequent requests
	BufPtr	Pointer to the granted buffer
<b>Return value</b>	BufReq_ReturnType	BUFREQ_OK: success BUFREQ_E_NOT_OK: default error detected BUFREQ_E_BUSY: all buffers in use BUFREQ_E_OVFL: requested buffer too large
<b>Description</b>	Provides access to a transmit buffer of the specified controller	
<b>Available via</b>	WEth.h	

]()

**[SWS\_WEth\_00078]** [The function shall provide a transmit buffer resource. The Wireless Ethernet Driver shall lock the buffer until it receives a subsequent call of WEth\_Transmit service with the buffer index returned in the BufIdxPtr parameter.]()

**[SWS\_WEth\_00137]** [All locked transmit buffers shall be released if the controller is disabled via WEth\_SetControllerMode.]()

**[SWS\_WEth\_00079]** [If a buffer requested with WEth\_ProvideTxBuffer that is larger than the available buffer length, the buffer shall not be locked but return the available length and BUFREQ\_E\_OVFL.]()

**[SWS\_WEth\_00080]** [If all available buffers are in use the component shall return BUFREQ\_E\_BUSY.]()

**[SWS\_WEth\_00081]** [If development error detection is enabled: the function shall check that the service WEth\_Init was previously called. If the check fails, the function shall raise the development error WETH\_E\_UNINIT.]()

**[SWS\_WEth\_00082]** [If development error detection is enabled: the function shall check the parameter `CtrlIdx` for being valid. If the check fails, the function shall raise the development error `WETH_E_INV_CTRL_IDX`.]()

**[SWS\_WEth\_00083]** [If development error detection is enabled: the function shall check the parameter `BufIdxPtr` for being valid. If the check fails, the function shall raise the development error `WETH_E_PARAM_POINTER`.]()

**[SWS\_WEth\_00084]** [If development error detection is enabled: the function shall check the parameter `BufPtr` for being valid. If the check fails, the function shall raise the development error `WETH_E_PARAM_POINTER`.]()

**[SWS\_WEth\_00085]** [If development error detection is enabled: the function shall check the parameter `LenBytePtr` for being valid. If the check fails, the function shall raise the development error `WETH_E_PARAM_POINTER`.]()

**[SWS\_WEth\_00086]** [Caveat: The function requires previous controller initialization (`WEth_Init`).]()

### 8.3.8 WEth\_Transmit

**[SWS\_WEth\_00087]** Definition of API function `WEth_Transmit` [

<b>Service Name</b>	WEth_Transmit	
<b>Syntax</b>	<pre>Std_ReturnType WEth_Transmit (     uint8 CtrlId,     Eth_BufIdxType BufId,     Eth_FrameType FrameType,     boolean TxConfirmation,     uint16 LenByte,     const uint8* PhysAddrPtr )</pre>	
<b>Service ID [hex]</b>	0x14	
<b>Sync/Async</b>	Synchronous	
<b>Reentrancy</b>	Non Reentrant	
<b>Parameters (in)</b>	<code>CtrlId</code>	Index of the context within the Wireless Ethernet Driver
	<code>BufId</code>	Index of the buffer resource
	<code>FrameType</code>	Ethernet frame type
	<code>TxConfirmation</code>	Activates transmission confirmation
	<code>LenByte</code>	Data length in byte (802.11 Header + Body, not including FCS)
	<code>PhysAddrPtr</code>	Physical target address (MAC address) in network byte order
<b>Parameters (inout)</b>	None	
<b>Parameters (out)</b>	None	
<b>Return value</b>	<code>Std_ReturnType</code>	<code>E_OK</code> : success <code>E_NOT_OK</code> : transmission failed
<b>Description</b>	Triggers transmission of a previously filled transmit buffer	
<b>Available via</b>	WEth.h	

]()

**[SWS\_WEth\_00088]** [The function shall build the Ethernet header with the given physical target address (MAC address) and trigger the transmission of a previously filled transmit buffer.]()

After transmission, the driver needs to release the allocated buffer. It is up to the implementation when the actual buffer release shall occur, e.g. within the context of the WEth\_TxConfirmation, the WEth\_MainFunction, or during the next WEth\_ProvideTxBuffer.

**[SWS\_WEth\_00138]** [All pending transmit buffers shall be released if the controller is disabled via WEth\_SetControllerMode.]()

**[SWS\_WEth\_00090]** [If development error detection is enabled: the function shall check that the service WEth\_Init was previously called. If the check fails, the function shall raise the development error WETH\_E\_UNINIT.]()

**[SWS\_WEth\_00091]** [If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error WETH\_E\_INV\_CTRL\_IDX.]()

**[SWS\_WEth\_00092]** [If development error detection is enabled: the function shall check the parameter BufIdx for being valid. If the check fails, the function shall raise the development error WETH\_E\_INV\_PARAM.]()

**[SWS\_WEth\_00093]** [If development error detection is enabled: the function shall check the parameter PhysAddrPtr for being valid. If the check fails, the function shall raise the development error WETH\_E\_PARAM\_POINTER.]()

**[SWS\_WEth\_00129]** [If development error detection is enabled: the function shall check the controller mode for being active (ETH\_MODE\_ACTIVE). If the check fails, the function shall raise the development error WETH\_E\_INV\_MODE.]()

**[SWS\_WEth\_00094]** [Caveat: The function requires previous buffer request (WEth\_ProvideTxBuffer).]()

### 8.3.9 WEth\_TxConfirmation

**[SWS\_WEth\_00100] Definition of API function WEth\_TxConfirmation** [

<b>Service Name</b>	WEth_TxConfirmation	
<b>Syntax</b>	void WEth_TxConfirmation ( uint8 CtrlId )	
<b>Service ID [hex]</b>	0x02	
<b>Sync/Async</b>	Synchronous	
<b>Reentrancy</b>	Non Reentrant	
<b>Parameters (in)</b>	CtrlId	Index of the controller within the context of the Wireless Ethernet Driver





<b>Parameters (inout)</b>	None
<b>Parameters (out)</b>	None
<b>Return value</b>	None
<b>Description</b>	Triggers frame transmission confirmation
<b>Available via</b>	WEth.h

]()

**[SWS\_WEth\_00101]** [The function shall check all filled transmit buffers for successful transmission. The function issues transmit confirmation for each transmitted frame using the callback function WEthIf\_TxConfirmation if requested by the previous call of WEth\_Transmit service.]()

**[SWS\_WEth\_00102]** [If transmission confirmation was enabled by a previous call to WEth\_Transmit function the function shall release the buffer resource.]()

**[SWS\_WEth\_00103]** [If development error detection is enabled: the function shall check that the service WEth\_Init was previously called. If the check fails, the function shall raise the development error WETH\_E\_UNINIT.]()

**[SWS\_WEth\_00104]** [If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error WETH\_E\_INV\_CTRL\_IDX.]()

**[SWS\_WEth\_00134]** [If development error detection is enabled: the function shall check the controller mode for being active (ETH\_MODE\_ACTIVE). If the check fails, the function shall raise the development error WETH\_E\_INV\_MODE.]()

**[SWS\_WEth\_00105]** [Caveat: The function requires previous initialization (WEth\_Init).]()

**[SWS\_WEth\_10063]** [The module must ensure that within the interrupt/polling context of this function call, transmission parameters of the wireless channel for the current buffer could be retrieved by the function WEth\_GetBufWTxParams.]()

### 8.3.10 WEth\_Receive

**[SWS\_WEth\_00095] Definition of API function WEth\_Receive** [

<b>Service Name</b>	WEth_Receive
<b>Syntax</b>	<pre>void WEth_Receive (     uint8 CtrlId,     Eth_RxStatusType* RxStatusPtr )</pre>
<b>Service ID [hex]</b>	0x05
<b>Sync/Async</b>	Synchronous
<b>Reentrancy</b>	Non Reentrant







<b>Parameters (in)</b>	CtrlId	Index of the context within the Wireless Ethernet Driver
<b>Parameters (inout)</b>	None	
<b>Parameters (out)</b>	RxStatusPtr	Indicates whether a frame has been received and if so, whether more frames are available or frames got lost.
<b>Return value</b>	void	–
<b>Description</b>	Triggers frame reception.	
<b>Available via</b>	WEth.h	

]()

**[SWS\_WEth\_00096]** [The function shall read the next frame from the receive buffers. The function passes the received frame to the Ethernet interface using the callback function WEthIf\_RxIndication and indicates if there are more frames in the receive buffers.]()

**[SWS\_WEth\_00097]** [If development error detection is enabled: the function shall check that the service WEth\_Init was previously called. If the check fails, the function shall raise the development error WETH\_E\_UNINIT.]()

**[SWS\_WEth\_00098]** [If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error WETH\_E\_INV\_CTRL\_IDX.]()

**[SWS\_WEth\_00132]** [If development error detection is enabled: the function shall check the controller mode for being active (ETH\_MODE\_ACTIVE). If the check fails, the function shall raise the development error WETH\_E\_INV\_MODE.]()

**[SWS\_WEth\_00153]** [When calling the callback function WEthIf\_RxIndication broadcast frames shall be indicated to the Ethernet Interface (see [9]).]()

**[SWS\_WEth\_00099]** [Caveat: The function requires previous controller initialization (WEth\_Init).]()

**[SWS\_WEth\_10061]** [The module must ensure that within the interrupt/polling context of this function call, reception parameters of the wireless channel for the current buffer could be retrieved by the function WEth\_GetBufWRxParams.]()

### 8.3.11 WEth\_GetWETHERStats32

**[SWS\_WEth\_10070]** Definition of API function WEth\_GetWETHERStats32 [

<b>Service Name</b>	WEth_GetWETHERStats32
<b>Syntax</b>	Std_ReturnType WEth_GetWETHERStats32 ( uint8 CtrlId, uint32* WETHERStats )
<b>Service ID [hex]</b>	0x15





<b>Sync/Async</b>	Synchronous	
<b>Reentrancy</b>	Non Reentrant	
<b>Parameters (in)</b>	CtrlId	Index of the context within the Wireless Ethernet driver
<b>Parameters (inout)</b>	None	
<b>Parameters (out)</b>	WEtherStats	List of values according to IEEE 802.11-2012
<b>Return value</b>	Std_ReturnType	E_OK: success E_NOT_OK: drop counter could not be obtained
<b>Description</b>	Returns the following list according to IEEE 802.11-2012, where the maximal possible value shall denote an invalid value, e.g. if this counter is not available: 1. dot11STAStatistics TransmittedFragmentCount 2. dot11STAStatisticsGroupTransmittedFrameCount 3. dot11STAStatisticsFailedCount 4. dot11STAStatisticsRetryCount 5. dot11STAStatisticsMultipleRetryCount 6. dot11STAStatisticsFrameDuplicateCount 7. dot11STAStatisticsRTSSuccessCount 8. dot11STAStatisticsRTSFailureCount 9. dot11STAStatisticsACKFailureCount 10. dot11STAStatisticsQosTransmittedFragmentCount 11. dot11STAStatisticsQosFailedCount 12. dot11STAStatisticsQosRetryCount 13. dot11STAStatisticsQosMultipleRetryCount 14. dot11STAStatisticsQosFrameDuplicateCount 15. dot11STAStatisticsQosRTSSuccessCount 16. dot11STAStatisticsQosRTSFailureCount 17. dot11STAStatisticsQosACKFailureCount 18. dot11STAStatisticsQosReceivedFragmentCount 19. dot11STAStatisticsQosTransmittedFrameCount 20. dot11STAStatisticsQosDiscardedFrameCount 21. dot11STAStatisticsQosMPDUsReceivedCount 22. dot11STAStatisticsQosRetriesReceivedCount 23. dot11STAStatisticsReceivedFragmentCount 24. dot11STAStatisticsGroupReceivedFrameCount 25. dot11STAStatisticsFCSErrorCount 26. dot11STAStatisticsTransmittedFrameCount 27. dot11STAStatisticsRSNAStatsCMACICVErrors 28. dot11STAStatisticsRSNAStatsCMACReplays 29. dot11STAStatisticsRSNAStatsRobustMgmtCCMPReplays 30. dot11STAStatisticsRSNAStatsTKIPICVErrors 31. dot11STAStatisticsRSNAStatsTKIPReplays 32. dot11STAStatisticsRSNAStatsCCMPDecryptErrors 33. dot11STAStatisticsRSNAStatsCCMPReplays 34. dot11STAStatisticsTransmittedAMSDUCount 35. dot11STAStatisticsFailedAMSDUCount 36. dot11STAStatisticsRetryAMSDUCount 37. dot11STAStatisticsMultipleRetryAMSDUCount 38. dot11STAStatisticsAMSDUAckFailureCount 39. dot11STAStatisticsReceivedAMSDUCount 40. dot11STAStatisticsTransmittedAMPDUCount 41. dot11STAStatisticsTransmittedMPDUsInAMPDUCount 42. dot11STAStatisticsAMPDUReceivedCount 43. dot11STAStatisticsMPDUInReceivedAMPDUCount 44. dot11STAStatisticsAMPDUDelimiterCRCErrorCount 45. dot11STAStatisticsImplicitBARFailureCount 46. dot11STAStatisticsExplicitBARFailureCount 47. dot11STAStatisticsChannelWidthSwitchCount 48. dot11STAStatisticsTwentyMHzFrameTransmittedCount 49. dot11STAStatisticsFortyMHzFrameTransmittedCount 50. dot11STAStatisticsTwentyMHzFrameReceivedCount 51. dot11STAStatisticsFortyMHzFrameReceivedCount 52. dot11STAStatisticsPSMPUTTGrantDuration 53. dot11STAStatisticsPSMPUTTUsedDuration 54. dot11STAStatisticsGrantedRDGUsedCount 55. dot11STAStatisticsGrantedRDGUnusedCount 56. dot11STAStatisticsTransmittedFramesInGrantedRDGCount 57. dot11STAStatisticsDualCTSSuccessCount 58. dot11STAStatisticsDualCTSFailureCount 59. dot11STAStatisticsRTSLSIGSuccessCount 60. dot11STAStatisticsRTSLSIGFailureCount 61. dot11STAStatisticsBeamformingFrameCount 62. dot11STAStatisticsSTBCCTSSuccessCount 63. dot11STAStatisticsSTBCCTSFailureCount 64. dot11STAStatisticsnonSTBCCTSSuccessCount 65. dot11STAStatisticsnonSTBCCTSFailureCount	
<b>Available via</b>	WEth.h	

]() Note: Only Counter32 values from the list Dot11STAStatisticsReportEntry in 802.11-2012 (C.3) are supported.

**[SWS\_WEth\_00234]** [The function shall read a list of values from the indexed controller according to [10].]()

**[SWS\_WEth\_00235]** [If development error detection is enabled: the function shall check that the service WEth\_Init was previously called. If the check fails, the function shall raise the development error WETH\_E\_UNINIT.]()

**[SWS\_WEth\_00236]** [If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error WETH\_E\_INV\_CTRL\_IDX.]()

**[SWS\_WEth\_00237]** [If development error detection is enabled: the function shall check the parameter RxStats for being valid. If the check fails, the function shall raise the development error WETH\_E\_PARAM\_POINTER.]()

**[SWS\_WEth\_00238]** [The function WEth\_GetWEtherStats32 shall be pre compile time configurable On/Off by the configuration parameter: WEthGetWEtherStatsApi.]()

### 8.3.12 WEth\_GetWEtherStats64

**[SWS\_WEth\_10024]** Definition of API function WEth\_GetWEtherStats64 [

<b>Service Name</b>	WEth_GetWEtherStats64	
<b>Syntax</b>	Std_ReturnType WEth_GetWEtherStats64 ( uint8 CtrlId, uint64* WEtherStats )	
<b>Service ID [hex]</b>	0xe0	
<b>Sync/Async</b>	Synchronous	
<b>Reentrancy</b>	Non Reentrant	
<b>Parameters (in)</b>	CtrlId	Index of the context within the Wireless Ethernet driver
<b>Parameters (inout)</b>	None	
<b>Parameters (out)</b>	WEtherStats	List of values according to IEEE 802.11-2012
<b>Return value</b>	Std_ReturnType	E_OK: success E_NOT_OK: drop counter could not be obtained
<b>Description</b>	Returns the following list according to IEEE 802.11-2012, where the maximal possible value shall denote an invalid value, e.g. if this counter is not available: 1. dot11STAStatisticsTransmittedOctetsInAMSDUCount 2. dot11STAStatisticsReceivedOctetsInAMSDUCount 3. dot11STAStatisticsTransmittedOctetsInAMPDUCount 4. dot11STAStatisticsReceivedOctetsInAMPDUCount 5. dot11STAStatisticsTransmittedOctetsInGrantedRDGCount	
<b>Available via</b>	WEth.h	

]() Note: Only Counter64 values from the list Dot11STAStatisticsReportEntry in 802.11-2012 (C.3) are supported.

**[SWS\_WEth\_10026]** [The function shall read a list of values from the indexed controller according to [10].]()

**[SWS\_WEth\_10235]** [If development error detection is enabled: the function shall check that the service WEth\_Init was previously called. If the check fails, the function shall raise the development error WETH\_E\_UNINIT.]()

**[SWS\_WEth\_10236]** [If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error WETH\_E\_INV\_CTRL\_IDX.]()

**[SWS\_WEth\_10237]** [If development error detection is enabled: the function shall check the parameter RxStats for being valid. If the check fails, the function shall raise the development error WETH\_E\_PARAM\_POINTER.]()

**[SWS\_WEth\_10027]** [The function WEth\_GetWEtherStats64 shall be pre compile time configurable On/Off by the configuration parameter: WEthGetWEtherStatsApi.]  
( )

### 8.3.13 WEth\_WriteTrcvRegs

**[SWS\_WEth\_10028]** Definition of API function WEth\_WriteTrcvRegs [

<b>Service Name</b>	WEth_WriteTrcvRegs	
<b>Syntax</b>	<pre>Std_ReturnType WEth_WriteTrcvRegs (     uint8 CtrlId,     uint8 TrcvId,     uint8 RadioId,     const uint32* RegIds,     const uint32* RegVals,     uint8 NumRegs )</pre>	
<b>Service ID [hex]</b>	0x30	
<b>Sync/Async</b>	Synchronous	
<b>Reentrancy</b>	Non Reentrant	
<b>Parameters (in)</b>	CtrlId	Index of the controller within the context of the Ethernet Driver
	TrcvId	Index of the transceiver on the destined bus
	RadioId	Index of the Transceiver's Radio Module
	RegIds	List of Index of the transceiver registers
	RegVals	Value to be written into the indexed register
	NumRegs	Number of Registers/Values
<b>Parameters (inout)</b>	None	
<b>Parameters (out)</b>	None	
<b>Return value</b>	Std_ReturnType	E_OK: Service accepted E_NOT_OK: Service denied
<b>Description</b>	Configures a transceivers registers or triggers a function offered by the receiver	
<b>Available via</b>	WEth.h	

]()

**[SWS\_WEth\_00059]** [The function shall write the specified parameters in the transceivers registers for the indexed radio through a controller specific bus interface of the indexed controller.]()

**[SWS\_WEth\_00060]** [If development error detection is enabled: the function shall check that the service WEth\_Init was previously called. If the check fails, the function shall raise the development error WETH\_E\_UNINIT.]()

**[SWS\_WEth\_00061]** [If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error WETH\_E\_INV\_CTRL\_IDX.]()

**[SWS\_WEth\_00063]** [Caveat: The function requires previous controller initialization (WEth\_Init).]()

**[SWS\_WEth\_10030]** [If development error detection is enabled: the function shall check the parameter RegIds for being valid. If the check fails, the function shall raise the development error WETH\_E\_PARAM\_POINTER.]()

**[SWS\_WEth\_10031]** [If development error detection is enabled: the function shall check the parameter RegVals for being valid. If the check fails, the function shall raise the development error WETH\_E\_PARAM\_POINTER.]()

### 8.3.14 WEth\_ReadTrcvRegs

**[SWS\_WEth\_10032] Definition of API function WEth\_ReadTrcvRegs** [

<b>Service Name</b>	WEth_ReadTrcvRegs	
<b>Syntax</b>	<pre>Std_ReturnType WEth_ReadTrcvRegs (     uint8 CtrlId,     uint8 TrcvId,     uint8 RadioId,     const uint32* RegIds,     uint32* RegValsPtr,     uint8 NumRegs )</pre>	
<b>Service ID [hex]</b>	0x31	
<b>Sync/Async</b>	Synchronous	
<b>Reentrancy</b>	Non Reentrant	
<b>Parameters (in)</b>	CtrlId	Index of the controller within the context of the Ethernet Driver
	TrcvId	Index of the transceiver on the destined bus
	RadioId	Index of the Transceiver's Radio Module
	RegIds	Array of Index of the transceiver registers
	NumRegs	Number of Registers/Values
<b>Parameters (inout)</b>	None	
<b>Parameters (out)</b>	RegValsPtr	Value to be written into the indexed register
<b>Return value</b>	Std_ReturnType	E_OK: Service accepted E_NOT_OK: Service denied
<b>Description</b>	Reads a transceiver register	
<b>Available via</b>	WEth.h	

]()

**[SWS\_WEth\_00065]** [The function shall read the specified transceiver register through the MII of the indexed controller.]()

**[SWS\_WEth\_00066]** [If development error detection is enabled: the function shall check that the service WEth\_Init was previously called. If the check fails, the function shall raise the development error WETH\_E\_UNINIT.]()

**[SWS\_WEth\_00067]** [If development error detection is enabled: the function shall check the parameter CtrlIdx for being valid. If the check fails, the function shall raise the development error WETH\_E\_INV\_CTRL\_IDX.]()

**[SWS\_WEth\_00068]** [If development error detection is enabled: the function shall check the parameter RegValPtr for being valid. If the check fails, the function shall raise the development error WETH\_E\_PARAM\_POINTER.]()

**[SWS\_WEth\_00070]** [Caveat: The function requires previous controller initialization (WEth\_Init).]()

**[SWS\_WEth\_10034]** [If development error detection is enabled: the function shall check the parameter RegIds for being valid. If the check fails, the function shall raise the development error WETH\_E\_PARAM\_POINTER.]()

**[SWS\_WEth\_10035]** [If development error detection is enabled: the function shall check the parameter RegVals for being valid. If the check fails, the function shall raise the development error WETH\_E\_PARAM\_POINTER.]()

### 8.3.15 WEth\_GetBufWRxParams

**[SWS\_WEth\_10062]** Definition of API function WEth\_GetBufWRxParams [

<b>Service Name</b>	WEth_GetBufWRxParams	
<b>Syntax</b>	<pre>Std_ReturnType WEth_GetBufWRxParams (     uint8 CtrlId,     const WEth_BufWRxParamIdType* RxParamIds,     uint32* ParamValues,     uint8 NumParams )</pre>	
<b>Service ID [hex]</b>	0x34	
<b>Sync/Async</b>	Synchronous	
<b>Reentrancy</b>	Non Reentrant	
<b>Parameters (in)</b>	CtrlId	Index of the Ethernet controller
	RxParamIds	IDs of the Parameter that are requested
	NumParams	Number of Parameters that are requested
<b>Parameters (inout)</b>	None	
<b>Parameters (out)</b>	ParamValues	Values of the Parameters requested
<b>Return value</b>	Std_ReturnType	E_OK: success E_NOT_OK: failed reading parameters
<b>Description</b>	Read out values related to the receive direction for a received packet. For example, this could be RSSI or Channel belonging to one single packet. This API is valid only within the context of WEth_Receive	
<b>Available via</b>	WEth.h	

]()

**[SWS\_WEth\_10039]** [If development error detection is enabled: the function shall check that the service WEth\_Init was previously called. If the check fails, the function shall raise the development error WETH\_E\_UNINIT.]([SRS\\_BSW\\_00487](#))

**[SWS\_WEth\_10040]** [If development error detection is enabled: the function shall check the parameter CtrlId for being valid. If the check fails, the function shall raise the development error WETH\_E\_INV\_CTRL\_ID.]()

**[SWS\_WEth\_10041]** [If development error detection is enabled: the function shall check the parameter RxParamIds for being valid. If the check fails, the function shall raise the development error WETH\_E\_PARAM\_POINTER.]()

**[SWS\_WEth\_10042]** [If development error detection is enabled: the function shall check the parameter ParamValues for being valid. If the check fails, the function shall raise the development error WETH\_E\_PARAM\_POINTER.]()

### 8.3.16 WEth\_GetBufWTxParams

**[SWS\_WEth\_10044] Definition of API function WEth\_GetBufWTxParams [**

<b>Service Name</b>	WEth_GetBufWTxParams	
<b>Syntax</b>	<pre>Std_ReturnType WEth_GetBufWTxParams (     uint8 CtrlId,     const WEth_BufWTxParamIdType* TxParamIds,     uint32* ParamValues,     uint8 NumParams )</pre>	
<b>Service ID [hex]</b>	0x35	
<b>Sync/Async</b>	Synchronous	
<b>Reentrancy</b>	Non Reentrant	
<b>Parameters (in)</b>	CtrlId	Index of the Ethernet controller
	TxParamIds	IDs of the Parameter that are requested
	NumParams	Number of Parameters that are requested
<b>Parameters (inout)</b>	None	
<b>Parameters (out)</b>	ParamValues	Values of the Parameters requested
<b>Return value</b>	Std_ReturnType	E_OK: success E_NOT_OK: failed reading parameters
<b>Description</b>	Read out values related to the transmit direction for a transmitted packet. This API is valid only within the context of WEth_TxConfirmation.	
<b>Available via</b>	WEth.h	

]()

**[SWS\_WEth\_10046]** [If development error detection is enabled: the function shall check that the service WEth\_Init was previously called. If the check fails, the function shall raise the development error WETH\_E\_UNINIT.]([SRS\\_BSW\\_00487](#))

**[SWS\_WEth\_10047]** [If development error detection is enabled: the function shall check the parameter CtrlId for being valid. If the check fails, the function shall raise the development error WETH\_E\_INV\_CTRL\_ID.]()

**[SWS\_WEth\_10048]** [If development error detection is enabled: the function shall check the parameter TxParamIds for being valid. If the check fails, the function shall raise the development error WETH\_E\_PARAM\_POINTER.]()

**[SWS\_WEth\_10049]** [If development error detection is enabled: the function shall check the parameter ParamValues for being valid. If the check fails, the function shall raise the development error WETH\_E\_PARAM\_POINTER.]()

### 8.3.17 WEth\_SetBufWTxParams

#### [SWS\_WEth\_10051] Definition of API function WEth\_SetBufWTxParams [

<b>Service Name</b>	WEth_SetBufWTxParams	
<b>Syntax</b>	<pre>Std_ReturnType WEth_SetBufWTxParams (     uint8 CtrlId,     Eth_BufIdxType BufId,     const WEth_BufWTxParamIdType* TxParamIds,     const uint32* ParamValues,     uint8 NumParams )</pre>	
<b>Service ID [hex]</b>	0x36	
<b>Sync/Async</b>	Synchronous	
<b>Reentrancy</b>	Non Reentrant	
<b>Parameters (in)</b>	CtrlId	Index of the Ethernet controller
	BufId	Index of the buffer resource
	TxParamIds	IDs of the Parameter that are provided to the transmit radio
	ParamValues	Values of the Parameters that are provided to the transmit radio
	NumParams	Number of Parameters that are provided to the transmit radio
<b>Parameters (inout)</b>	None	
<b>Parameters (out)</b>	None	
<b>Return value</b>	Std_ReturnType	E_OK: success E_NOT_OK: failed setting parameter
<b>Description</b>	Set values related to the transmit direction for a specific buffer (packet to be sent). For example, this can be the desired transmit power or the channel belonging to one single packet.	
<b>Available via</b>	WEth.h	

]([SRS\\_V2X\\_00245](#))

**[SWS\_WEth\_10053]** [If development error detection is enabled: the function shall check that the service WEth\_Init was previously called. If the check fails, the function shall raise the development error WETH\_E\_NOT\_INITIALIZED.]()

**[SWS\_WEth\_10054]** [If development error detection is enabled: the function shall check the parameter CtrlId for being valid. If the check fails, the function shall raise the development error WETH\_E\_INV\_CTRL\_ID.]()

**[SWS\_WEth\_10055]** [If development error detection is enabled: the function shall check the parameter BufId for being valid. If the check fails, the function shall raise the development error WETH\_E\_INV\_PARAM.]()

**[SWS\_WEth\_10056]** [If development error detection is enabled: the function shall check the parameter TxParamIds for being valid. If the check fails, the function shall raise the development error WETH\_E\_PARAM\_POINTER.]()

**[SWS\_WEth\_10057]** [If development error detection is enabled: the function shall check the parameter ParamValues for being valid. If the check fails, the function shall raise the development error WETH\_E\_PARAM\_POINTER.]()



### 8.3.18 WEth\_GetVersionInfo

#### [SWS\_WEth\_00106] Definition of API function WEth\_GetVersionInfo [

<b>Service Name</b>	WEth_GetVersionInfo	
<b>Syntax</b>	<pre>void WEth_GetVersionInfo (     Std_VersionInfoType* VersionInfoPtr )</pre>	
<b>Service ID [hex]</b>	0x0d	
<b>Sync/Async</b>	Synchronous	
<b>Reentrancy</b>	Reentrant	
<b>Parameters (in)</b>	None	
<b>Parameters (inout)</b>	None	
<b>Parameters (out)</b>	VersionInfoPtr	Pointer to where to store the version information of this module.
<b>Return value</b>	None	
<b>Description</b>	Returns the version information of this module	
<b>Available via</b>	WEth.h	

]()

[SWS\_WEth\_00136] [If development error detection is enabled: the function shall check the parameter VersionInfoPtr for being valid. If the check fails, the function shall raise the development error WETH\_E\_PARAM\_POINTER.]()

### 8.3.19 WEth\_TriggerPriorityQueueTransmit

#### [SWS\_WEth\_10071] Definition of API function WEth\_TriggerPriorityQueueTransmit [

<b>Service Name</b>	WEth_TriggerPriorityQueueTransmit	
<b>Syntax</b>	<pre>Std_ReturnType WEth_TriggerPriorityQueueTransmit (     uint8 CtrlId,     uint8 PriorityQueue,     uint8 MaxTxPower )</pre>	
<b>Service ID [hex]</b>	0x37	
<b>Sync/Async</b>	Synchronous	
<b>Reentrancy</b>	Non Reentrant	
<b>Parameters (in)</b>	CtrlId	Index of the context within the Wireless Ethernet Driver
	PriorityQueue	Index of the Priority Queue
	MaxTxPower	Limit the Power of the packet in the Priority Queue
<b>Parameters (inout)</b>	None	
<b>Parameters (out)</b>	None	
<b>Return value</b>	Std_ReturnType	E_OK: success E_NOT_OK: transmission failed
<b>Description</b>	Triggers transmission of a previously filled transmit buffer that is waiting in a software priority queue.	
<b>Available via</b>	WEth.h	

]()

## 8.4 Callback notifications

The Wireless Ethernet Driver does not provide any callback functions.

## 8.5 Scheduled functions

### 8.5.1 WEth\_MainFunction

#### [SWS\_WEth\_00171] Definition of scheduled function WEth\_MainFunction [

<b>Service Name</b>	WEth_MainFunction
<b>Syntax</b>	void WEth_MainFunction ( void )
<b>Service ID [hex]</b>	0x0a
<b>Description</b>	Support for indirect transmissions (extended frame timing constraints) and mechanisms for channel selection when using multiple channels. Used for polling state changes. Calls Ethlf_CtrlModeIndication when the controller mode changed.
<b>Available via</b>	SchM_WEth.h

]()

## 8.6 Expected interfaces

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

### 8.6.1 Mandatory Interfaces

This chapter defines all external interfaces, which are required to fulfill the core functionality of the module.

#### [SWS\_WEth\_00119] Definition of mandatory interfaces in module WEth [

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 $\{Dem/DemConfigSet/DemEventParameter/DemEventReportingType\} == STANDARD\_REPORTING$
Ethlf_CtrlModeIndication	Ethlf.h	Called asynchronously when mode has been read out. Triggered by previous <EthDrv>_SetController Mode call. Can directly be called within the trigger functions.
Ethlf_GetVersionInfo	Ethlf.h	Returns the version information of this module





<i>API Function</i>	<i>Header File</i>	<i>Description</i>
Ethlf_MainFunctionRx	SchM_Ethlf.h	The function checks for new received frames and issues reception indications in polling mode.
Ethlf_MainFunctionTx	SchM_Ethlf.h	The function issues transmission confirmations in polling mode. It checks also for transceiver state changes.
Ethlf_RxIndication	Ethlf.h	Receive indication of an Ethernet frame which was received by the indexed controller
Ethlf_TxConfirmation	Ethlf.h	Confirms frame transmission by the indexed controller

]()

### 8.6.2 Optional Interfaces

This chapter defines all external interfaces, which are required to fulfill an optional functionality of the module.

#### [SWS\_WEth\_00120] Definition of optional interfaces in module WEth [

<i>API Function</i>	<i>Header File</i>	<i>Description</i>
Det_ReportError	Det.h	Service to report development errors.

]()

### 8.6.3 Configurable interfaces

The Wireless Ethernet Driver does not use configurable interfaces.

## 9 Sequence diagrams

The Wireless Ethernet Driver will interact with Ethernet Interface in the same way as the Ethernet Driver, see sequence diagrams in [9, SWS Ethernet Interface].

## 10 Configuration specification

Chapter 10.1 specifies the structure (containers) and the parameters of the WEth module.

Chapter 10.2 specifies additionally published information of the WEth module.

### 10.1 Containers and configuration parameters

The following chapters summarize all configuration parameters.

**[SWS\_WEth\_00040]** [The Wireless Ethernet Driver module shall reject configurations with partition mappings, which are not supported by the implementation.] ()

#### 10.1.1 Variants

No content.

### 10.2 WEth

<b>SWS Item</b>	<b>[ECUC_WEth_00037]</b>
<b>Module Name</b>	WEth
<b>Description</b>	Configuration of the WEth (Wireless Ethernet Driver) module.
<b>Post-Build Variant Support</b>	true
<b>Supported Config Variants</b>	VARIANT-LINK-TIME, VARIANT-POST-BUILD, VARIANT-PRE-COMPILE

Included Containers		
Container Name	Multiplicity	Scope / Dependency
<a href="#">WEthConfigSet</a>	1	This container contains the configuration parameters and sub containers of the AUTOSAR WEth module.
<a href="#">WEthGeneral</a>	1	General configuration of Wireless Ethernet Driver module.

### 10.3 WEthConfigSet

<b>SWS Item</b>	<b>[ECUC_WEth_00015]</b>
<b>Container Name</b>	WEthConfigSet
<b>Parent Container</b>	<a href="#">WEth</a>
<b>Description</b>	This container contains the configuration parameters and sub containers of the AUTOSAR WEth module.
<b>Configuration Parameters</b>	

Included Containers		
Container Name	Multiplicity	Scope / Dependency
WEthCtrlConfig	1..*	Configuration of the individual controller

## 10.4 WEthCtrlConfig

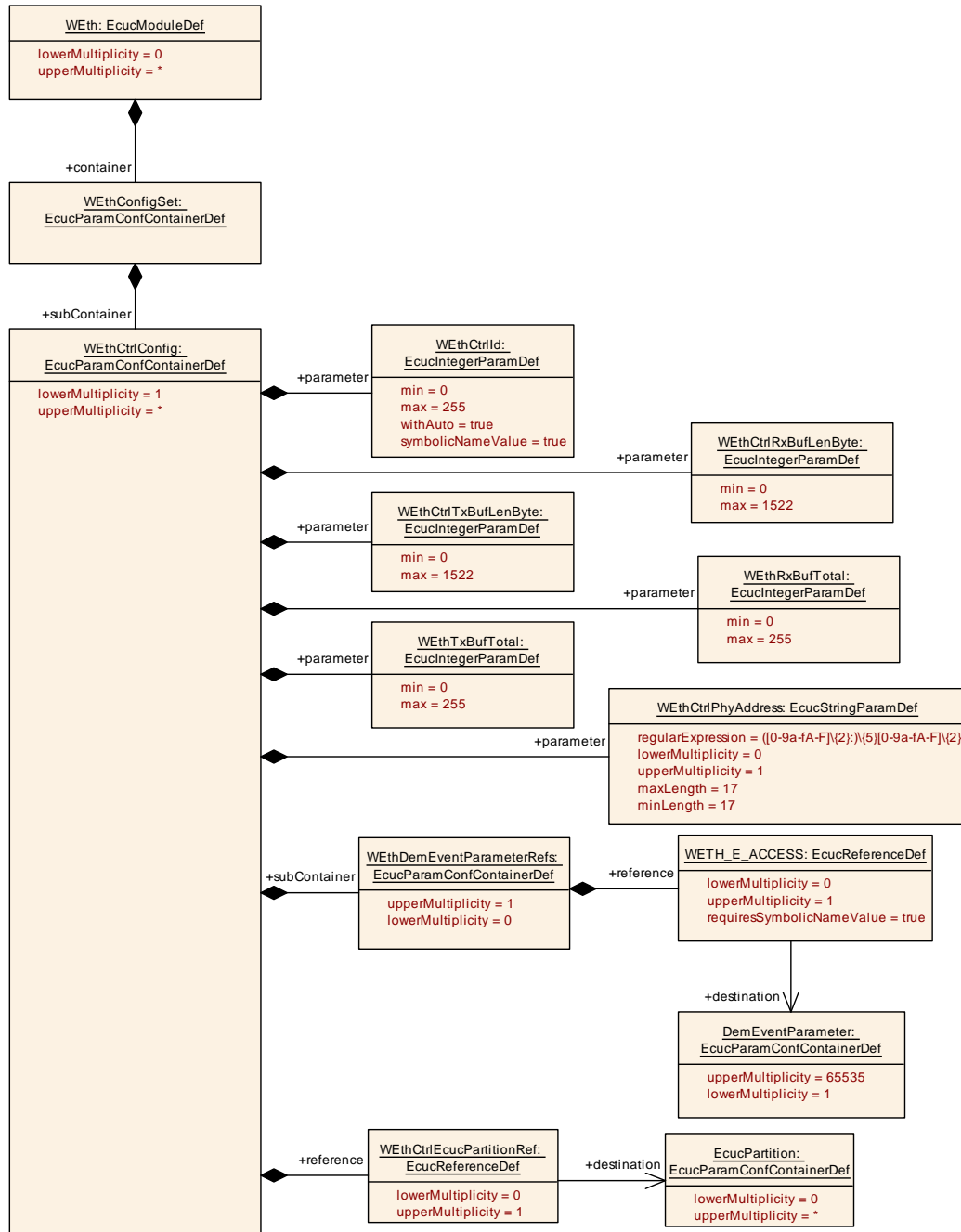


Figure 10.1: WEthCtrlConfig

<b>SWS Item</b>	<b>[ECUC_WEth_00006]</b>		
<b>Container Name</b>	WEthCtrlConfig		
<b>Parent Container</b>	<a href="#">WEthConfigSet</a>		
<b>Description</b>	Configuration of the individual controller		
<b>Configuration Parameters</b>			

<b>SWS Item</b>	<b>[ECUC_WEth_00007]</b>		
<b>Parameter Name</b>	WEthCtrlId		
<b>Parent Container</b>	<a href="#">WEthCtrlConfig</a>		
<b>Description</b>	Specifies the instance ID of the configured controller.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
<b>Range</b>	0 .. 255		
<b>Default value</b>	–		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	–	
	<b>Post-build time</b>	–	
<b>Scope / Dependency</b>	scope: ECU withAuto = true		

<b>SWS Item</b>	<b>[ECUC_WEth_00020]</b>		
<b>Parameter Name</b>	WEthCtrlPhyAddress		
<b>Parent Container</b>	<a href="#">WEthCtrlConfig</a>		
<b>Description</b>	Specifies the unique 48-bit physical address (MAC address) of the controller in network byte order.		
<b>Multiplicity</b>	0..1		
<b>Type</b>	EcucStringParamDef		
<b>Default value</b>	–		
<b>Length</b>	17-17		
<b>Regular Expression</b>	([0-9a-fA-F]{2}:){5}[0-9a-fA-F]{2}		
<b>Post-Build Variant Multiplicity</b>	true		
<b>Post-Build Variant Value</b>	true		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>[ECUC_WEth_00008]</b>		
<b>Parameter Name</b>	WEthCtrlRxBufLenByte		
<b>Parent Container</b>	<a href="#">WEthCtrlConfig</a>		
<b>Description</b>	Limits the maximum receive buffer length (frame length) in bytes.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucIntegerParamDef		





<b>Range</b>	0 .. 1522		
<b>Default value</b>	–		
<b>Post-Build Variant Value</b>	true		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>[ECUC_WEth_00009]</b>		
<b>Parameter Name</b>	WEthCtrlTxBufLenByte		
<b>Parent Container</b>	<a href="#">WEthCtrlConfig</a>		
<b>Description</b>	Limits the maximum transmit buffer length (frame length) in bytes.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucIntegerParamDef		
<b>Range</b>	0 .. 1522		
<b>Default value</b>	–		
<b>Post-Build Variant Value</b>	true		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>[ECUC_WEth_00013]</b>		
<b>Parameter Name</b>	WEthRxBufTotal		
<b>Parent Container</b>	<a href="#">WEthCtrlConfig</a>		
<b>Description</b>	Configures the number of receive buffers.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucIntegerParamDef		
<b>Range</b>	0 .. 255		
<b>Default value</b>	–		
<b>Post-Build Variant Value</b>	true		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>[ECUC_WEth_00014]</b>		
<b>Parameter Name</b>	WEthTxBufTotal		
<b>Parent Container</b>	<a href="#">WEthCtrlConfig</a>		
<b>Description</b>	Configures the number of transmit buffers.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucIntegerParamDef		
<b>Range</b>	0 .. 255		
<b>Default value</b>	–		
<b>Post-Build Variant Value</b>	true		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME







	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>[ECUC_WEth_00039]</b>		
<b>Parameter Name</b>	WEthCtrlEcucPartitionRef		
<b>Parent Container</b>	<a href="#">WEthCtrlConfig</a>		
<b>Description</b>	Maps the Wireless Ethernet controller to zero or one ECUC partitions. The ECUC partition referenced is a subset of the ECUC partitions where the Wireless Ethernet driver is mapped to.		
<b>Multiplicity</b>	0..1		
<b>Type</b>	Reference to EcucPartition		
<b>Post-Build Variant Multiplicity</b>	true		
<b>Post-Build Variant Value</b>	true		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	–	
	<b>Post-build time</b>	–	
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	–	
	<b>Post-build time</b>	–	
<b>Scope / Dependency</b>	scope: ECU		

Included Containers		
Container Name	Multiplicity	Scope / Dependency
<a href="#">WEthDemEventParameterRefs</a>	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.

## 10.5 WEthDemEventParameterRefs

<b>SWS Item</b>	<b>[ECUC_WEth_00016]</b>
<b>Container Name</b>	WEthDemEventParameterRefs
<b>Parent Container</b>	<a href="#">WEthCtrlConfig</a>
<b>Description</b>	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.
<b>Configuration Parameters</b>	

<b>SWS Item</b>	<b>[ECUC_WEth_00017]</b>
<b>Parameter Name</b>	WETH_E_ACCESS
<b>Parent Container</b>	<a href="#">WEthDemEventParameterRefs</a>
<b>Description</b>	Reference to the DemEventParameter which shall be issued when the error "Controller access failed" has occurred.

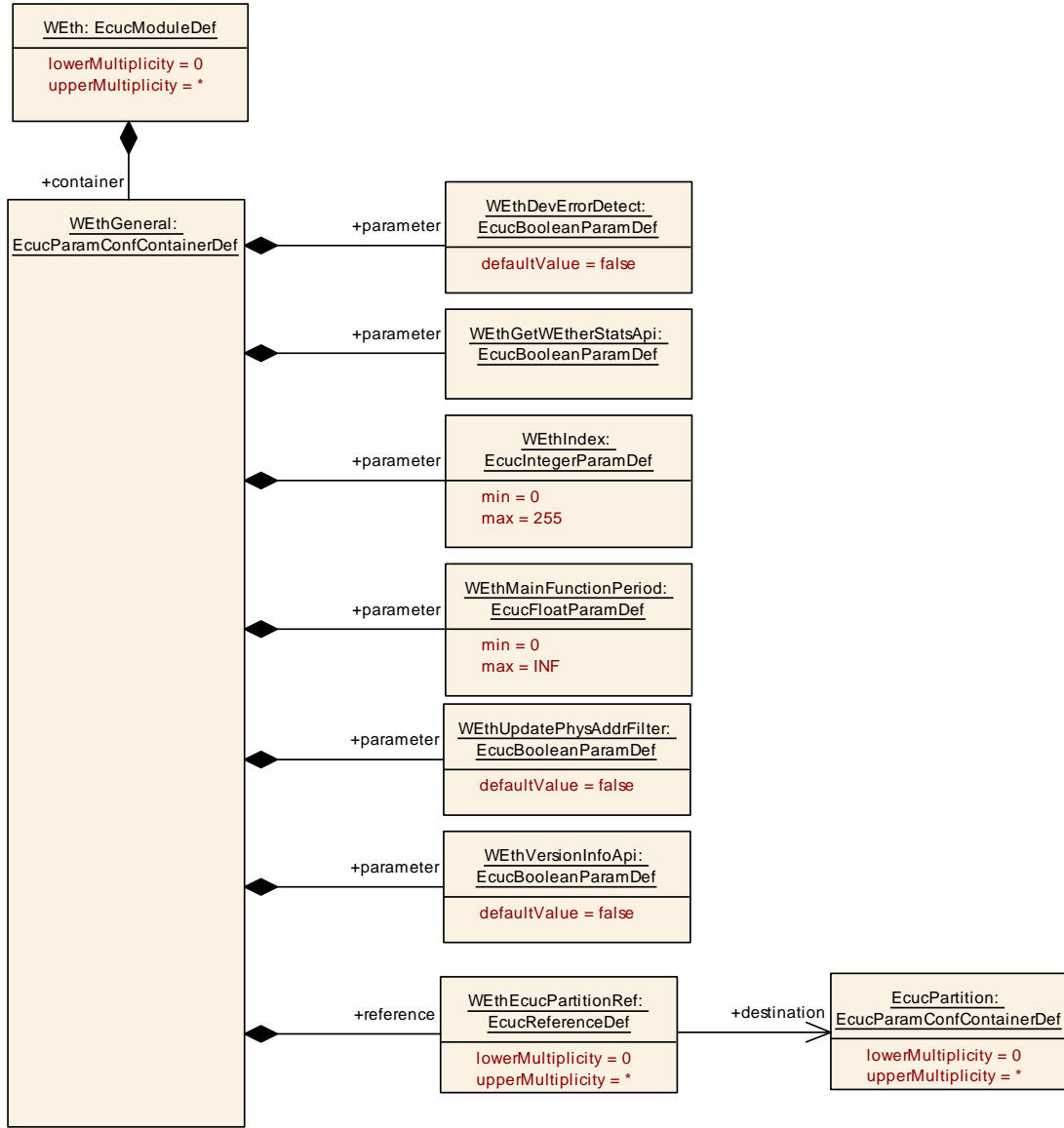


△

<b>Multiplicity</b>	0..1		
<b>Type</b>	Symbolic name reference to DemEventParameter		
<b>Post-Build Variant Multiplicity</b>	true		
<b>Post-Build Variant Value</b>	true		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

**No Included Containers**

### 10.6 WEthGeneral



**Figure 10.2: WEthGeneral**

<b>SWS Item</b>	[ECUC_WEth_00001]
<b>Container Name</b>	WEthGeneral
<b>Parent Container</b>	<a href="#">WEth</a>
<b>Description</b>	General configuration of Wireless Ethernet Driver module.
<b>Configuration Parameters</b>	

<b>SWS Item</b>	[ECUC_WEth_00003]
<b>Parameter Name</b>	WEthDevErrorDetect
<b>Parent Container</b>	<a href="#">WEthGeneral</a>





<b>Description</b>	Switches the Default Error Tracer (Det) detection and notification ON or OFF. <ul style="list-style-type: none"> <li>• true: detection and notification is enabled.</li> <li>• false: detection and notification is disabled.</li> </ul>		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	false		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	–	
	<b>Post-build time</b>	–	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>[ECUC_WEth_00036]</b>		
<b>Parameter Name</b>	WEthGetWEtherStatsApi		
<b>Parent Container</b>	<a href="#">WEthGeneral</a>		
<b>Description</b>	Enables / Disables WEth_GetWEtherStats_32 and WEth_GetWEtherStats_64 API.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	–		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	–	
	<b>Post-build time</b>	–	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>[ECUC_WEth_00018]</b>		
<b>Parameter Name</b>	WEthIndex		
<b>Parent Container</b>	<a href="#">WEthGeneral</a>		
<b>Description</b>	Specifies the InstanceId of this module instance. If only one instance is present it shall have the Id 0.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucIntegerParamDef		
<b>Range</b>	0 .. 255		
<b>Default value</b>	–		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	–	
	<b>Post-build time</b>	–	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>[ECUC_WEth_00022]</b>		
<b>Parameter Name</b>	WEthMainFunctionPeriod		
<b>Parent Container</b>	<a href="#">WEthGeneral</a>		
<b>Description</b>	Specifies the period of main function WEth_MainFunction in seconds. Wireless Ethernet driver does not require this information but the BSW scheduler.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucFloatParamDef		





<b>Range</b>	]0 .. INF[		
<b>Default value</b>	–		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	–	
	<b>Post-build time</b>	–	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>[ECUC_WEth_00019]</b>		
<b>Parameter Name</b>	WEthUpdatePhysAddrFilter		
<b>Parent Container</b>	<a href="#">WEthGeneral</a>		
<b>Description</b>	Enables/Disables optional API WEth_UpdatePhysAddrFilter.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	false		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	–	
	<b>Post-build time</b>	–	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>[ECUC_WEth_00004]</b>		
<b>Parameter Name</b>	WEthVersionInfoApi		
<b>Parent Container</b>	<a href="#">WEthGeneral</a>		
<b>Description</b>	Enables / Disables version info API.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	false		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	–	
	<b>Post-build time</b>	–	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>[ECUC_WEth_00038]</b>		
<b>Parameter Name</b>	WEthEcucPartitionRef		
<b>Parent Container</b>	<a href="#">WEthGeneral</a>		
<b>Description</b>	Maps the Wireless Ethernet driver to zero or multiple ECUC partitions to make the modules API available in this partition. The Wireless Ethernet driver will operate as an independent instance in each of the partitions.		
<b>Multiplicity</b>	0..*		
<b>Type</b>	Reference to EcucPartition		
<b>Post-Build Variant Multiplicity</b>	true		
<b>Post-Build Variant Value</b>	true		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	–	
	<b>Post-build time</b>	–	
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants



△

	<b>Link time</b>	-	
	<b>Post-build time</b>	-	
<b>Scope / Dependency</b>	scope: ECU		

No Included Containers

**[SWS\_WEth\_CONSTR\_00241]** [The module will operate as an independent instance in each of the partitions, means the called API will only target the partition it is called in.]()

**[SWS\_WEth\_CONSTR\_00242]** [If WEthEcucPartitionRef references one or more ECUC partitions, WEthCtrlEcucPartitionRef shall have a multiplicity of one and reference one of these ECUC partitions as well.]()

## A Change history of AUTOSAR traceable items

### A.1 Traceable item history of this document according to AUTOSAR Release R23-11

#### A.1.1 Added Specification Items in R23-11

none

#### A.1.2 Changed Specification Items in R23-11

Number	Heading
[SWS_WEth_00026]	Definition of imported datatypes of module WEth
[SWS_WEth_00119]	Definition of mandatory interfaces in module WEth
[SWS_WEth_10012]	Definition of datatype WEth_BufWRxParamIdType
[SWS_WEth_10013]	Definition of datatype WEth_BufWTxParamIdType
[SWS_WEth_10044]	Definition of API function WEth_GetBufWTxParams
[SWS_WEth_10066]	
[SWS_WEth_10068]	

**Table A.1: Changed Specification Items in R23-11**

#### A.1.3 Deleted Specification Items in R23-11

none

#### A.1.4 Added Constraints in R23-11

none

#### A.1.5 Changed Constraints in R23-11

none

#### A.1.6 Deleted Constraints in R23-11

none

## **A.2 Traceable item history of this document according to AUTOSAR Release R22-11**

### **A.2.1 Added Constraints in R22-11**

none

### **A.2.2 Changed Constraints in R22-11**

none

### **A.2.3 Deleted Constraints in R22-11**

none