Document Title	Specification of MACsec Key Agreement
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	1066

Document Status	published
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	R22-11

Document Change History			
Date	**Release**	**Changed by**	**Description**
2022-11-24	R22-11	AUTOSAR Release Management	• Initial release

Disclaimer

Contents

1 Introduction and functional overview

This specification describes the functionality, API and the configuration for the AUTOSAR Basic Software module MKA.

2 Acronyms and Abbreviations

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

Abbreviation / Acronym:	Description:
AN	Association Number
CA	Secure Connectivity Association
CAK	Secure Connectivity Association Key
DA	Destination Address
ICV	Integrity Check Value
KaY	MAC Security Key Agreement Entity
MACsec	Media Access Control Security
MKA	MACsec Key Agreement protocol (IEEE Std 802.1X)
MKPDU	MACsec Key Agreement Protocol Data Unit
MPDU	MACsec Protocol Data Unit
PAE	Port Access Entity
PN	Packet Number
SA	Secure Association or Source Address, as applicable
SAI	Secure Association Identifier
SAK	Secure Association Key
SC	Secure Channel
SCI	Secure Channel Identifier
SecTAG	MAC Security TAG
SecY	MAC Security Entity
SL	Short Length
SSCI	Short Secure Channel Identifier
TLV	Tag-Length-Value

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_TR_Glossary

[2] General Specification of Basic Software Modules
AUTOSAR_SWS_BSWGeneral

[3] Requirements on MACsec
AUTOSAR_RS_MACsec

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

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

[6] Advanced Encryption Standard (AES) Key Wrap Algorithm
https://tools.ietf.org/html/rfc3394

3.2 Related specification

AUTOSAR provides

- a General Specification on Basic Software modules [2, SWS BSW General], which is also valid for MKA.
- a MACsec Requirements Specification [3, RS MACsec] which is also valid for MKA.

Thus, the specification [2, SWS BSW General] shall be considered as additional and required specification for MKA.

4 Constraints and assumptions

4.1 Limitations

This document does not cover the integration neither requirements of the MACsec protocol as it is an IEEE published standard [4, IEEE 802.1AE-2018].

The AUTOSAR MACsec implementation currently has the following limitations:

- Only participants authentication based on Connectivity Association pre-shared keys (CAKs) is supported. (EAP-TLS, EAP-IKEv2, and other variants are not supported).
- Only MACsec between direct peers is supported (e.g. Point-to-Point configurations).
- Point-to-Multipoint configurations are not supported.
- In-service upgrades with EAPoL-MKA frames are not supported.
- Temporary suspension of MKA operation is not supported.
- MACsec Cipher Suites is the only currently supported EAPoL-Announcement TLV.
  The following EAPoL Announcements are currently not required:
  - Access Information → TLV Type 111
  - Key Management Domain → TLV Type 113
  - NID → TLV Type 114
- Dynamic Key Server election based on Key Server priority is not supported (Roles are set per configuration and fixed).
- The following MKPDU Parameter sets are currently not required:
  - Distributed CAK → Parameter set type 5
  - KMD → Parameter set type 6
  - ICV Indicator → Parameter set type 255

4.2 Applicability to car domains

Automotive systems require quicker start-up times for the devices connected to the on-board network, hence the protocol convergence time must be tuned accordingly.

5 Dependencies to other modules

The MACsec Key Agreement (MKA) Module has interfaces with the following modules:

1. EthIf → To configure, control, and monitor the MACsec Entity (per SW or HW).
2. CSM:
   - Protect outgoing MKA messages and validate incoming MKA messages.
   - Generate, encrypt, and decrypt session keys (SAKs).

5.1 Connection to Ethernet Interface (EthIf)

The MKA module and the EthIf are connected in order to:

- Receive and send MKA messages.
- Provide MACsec specific parameters to the lower layers.
- Orchestrate the Link-Up and Link-Down signaling of the interfaces for upper layers (i.e. through the EthSM).

In case an Ethernet Interface is MACsec protected, it will use a specific MKA module instance to configure the MACsec Entities (HW or SW) for transmission and reception.

In case the MACsec protected Ethernet Interface is required to be ACTIVE by the EthSM, after the signaling of the physical Link-up from the specific transceiver or Switch port(*ETH_MODE_ACTIVE*), the EthIf will delegate the establishment of at least one Secure Channel with the communication peers, to the referred MKA instance.
Once the MACsec Secure Channel is established and both participants can successfully receive and transmit, the Ethernet Interface will signal the Link-Up to the upper layers (e.g. through the Ethernet State Manager).

During the lifetime of the established SCs, the MKA module will maintain them alive by communicating with the MACsec Entities through the Ethernet Interface module. That means, updating the SC specific parameters in the MACsec Entities (Phy, Switch, or SW Entity).

Detailed information: The trigger to the MKA module to start the MACsec SC establishment is done after the EthTrcv or EthSwt mode indication to ACTIVE and before indicating this state to the EthSM (that means, the EthSM will stay in the *ETHSM_STATE_WAIT_TRCVLINK* state, as in this state the EthSM and the underlying EthIf is starting up the physical network interface, but the upper layer protocols (e.g., in TCP/IP) are not started yet).
Once triggered, the MKA module can start the needed actions to establish a MACsec

Secure Channel through the provided port. If MACsec is not configured in the port, the MKA module call will be skipped.

5.2 Indirect connection to EthDriver, EthSwitchDriver and EthTransceiverDriver

In case the MACsec Entity is offloaded to a HW device, the MKA module is indirectly connected to the EthDriver, EthSwitchDriver, and EthTranceiverDriver through the EthInterface. This connection is needed in order to establish, configure, and manage the needed MACsec Secure Channels. There are functions in the interface of the EthDriver, EthSwitchDriver and EthTransceiverDriver for that purpose.

Establishing a Secure Channel is done via the MACsec Key Agreement protocol, the MKA module will handle all protocol steps. These specific protocol datagrams are setup and organized by the MKA module. Thus, the MKA module provides via the existing function call the datagram to the Ethernet Interface, which then sends the datagram to the communication peer. This behavior is handled via a specific pair EtherType and message type, and is set via the interface. With this EtherType, the Ethernet Interface will handle the datagram on Rx and Tx trace.

5.3 Connection to Crypto Service Manager (CSM)

The MKA module requires a connection to the cryptographic BSW modules of AUTOSAR. This allows the MKA module to derive and use the needed keys and to interact with the cryptographic algorithms as specified in [5, IEEE-802.1X-2020] and [4, IEEE-802.1AE-2018].

For cryptographic usage, the MKA module needs following support from the BSW crypto:

- KDF (as described in [5, IEEE-802.1AE-2018] chapter 6.2.1) to derive ICK and KEK from CAK.
- AES-CMAC, which uses AES CMAC with 128bits, using ICK to generate and validate MKA message ICVs.
- A function to generate random data (for SAK and Member Identifier).
- AES-KEYWRAP based on [6, RFC 3394] to encrypt keys for transmission.

6 Requirements Tracing

The following tables reference the requirements specified in the documents listed in section 3.2 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_-00001]	MACsec Protocol support	[CP_SWS_Mka_00999]
[FO_RS_MACsec_-00002]	MACsec Key Agreement Protocol support	[CP_SWS_Mka_00001] [CP_SWS_Mka_00002] [CP_SWS_Mka_00003] [CP_SWS_Mka_00004] [CP_SWS_Mka_00005] [CP_SWS_Mka_00006] [CP_SWS_Mka_00007] [CP_SWS_Mka_00008] [CP_SWS_Mka_00009] [CP_SWS_Mka_00011] [CP_SWS_Mka_00015] [CP_SWS_Mka_00016] [CP_SWS_Mka_00017] [CP_SWS_Mka_00024] [CP_SWS_Mka_00031] [CP_SWS_Mka_00032] [CP_SWS_Mka_CONSTR_00019] [CP_SWS_Mka_CONSTR_00020]
[FO_RS_MACsec_-00003]	Using MACsec on external communication links	[CP_SWS_Mka_00001] [CP_SWS_Mka_00002] [CP_SWS_Mka_00006] [CP_SWS_Mka_00008] [CP_SWS_Mka_00011] [CP_SWS_Mka_00024]
[FO_RS_MACsec_-00004]	Configure which Ethernet ports use MACsec	[CP_SWS_Mka_00002]
[FO_RS_MACsec_-00005]	MACsec status control	[CP_SWS_Mka_00026] [CP_SWS_Mka_00027] [CP_SWS_Mka_00028] [CP_SWS_Mka_00029] [CP_SWS_Mka_00030]
[FO_RS_MACsec_-00006]	MACsec support for Adaptive AUTOSAR Platform	[CP_SWS_Mka_00999]
[FO_RS_MACsec_-00007]	Configuration of unprotected traffic (for Software-based MACsec)	[CP_SWS_Mka_00003] [CP_SWS_Mka_00004]
[FO_RS_MACsec_-00008]	Configuration of unprotected traffic (for Hardware-based MACsec)	[CP_SWS_Mka_00003] [CP_SWS_Mka_00004]
[FO_RS_MACsec_-00009]	MACsec Security Events	[CP_SWS_Mka_00025]
[FO_RS_MACsec_-00010]	Support of integrity and confidentiality	[CP_SWS_Mka_00008]
[FO_RS_MACsec_-00011]	MAC Security TAG	[CP_SWS_Mka_00999]
[FO_RS_MACsec_-00012]	MACsec EtherType	[CP_SWS_Mka_00999]
[FO_RS_MACsec_-00017]	Support of Extended Packet Number (XPN)	[CP_SWS_Mka_00008]
[FO_RS_MACsec_-00018]	Secure Channel Identifier (SCI)	[CP_SWS_Mka_00999]
[FO_RS_MACsec_-00019]	Secure Data	[CP_SWS_Mka_00999]
[FO_RS_MACsec_-00020]	Integrity Check Value (ICV)	[CP_SWS_Mka_CONSTR_00019] [CP_SWS_Mka_CONSTR_00020]
[FO_RS_MACsec_-00021]	Protect function in software solution	[CP_SWS_Mka_00999]
[FO_RS_MACsec_-00022]	Validation function in software solution	[CP_SWS_Mka_00999]

▽

△

Requirement	Description	Satisfied by
[FO_RS_MACsec_-00023]	Support of MKA Packets	[CP_SWS_Mka_00001] [CP_SWS_Mka_00008]
[FO_RS_MACsec_-00024]	Pre-shared key support	[CP_SWS_Mka_00001] [CP_SWS_Mka_00005] [CP_SWS_Mka_00016]
[FO_RS_MACsec_-00025]	Key selection via CKN	[CP_SWS_Mka_00001] [CP_SWS_Mka_00005] [CP_SWS_Mka_00006]
[FO_RS_MACsec_-00026]	GCM-based cipher support	[CP_SWS_Mka_CONSTR_00019] [CP_SWS_Mka_CONSTR_00020]
[FO_RS_MACsec_-00027]	Support of AES ciphers with at least 128 bits of key length	[CP_SWS_Mka_00009] [CP_SWS_Mka_CONSTR_00019] [CP_SWS_Mka_CONSTR_00020]
[FO_RS_MACsec_-00028]	Support of AES ciphers with 256 bits of key length	[CP_SWS_Mka_00009] [CP_SWS_Mka_CONSTR_00019] [CP_SWS_Mka_CONSTR_00020]
[FO_RS_MACsec_-00029]	Support of Key Encryption Key (KEK)	[CP_SWS_Mka_00001] [CP_SWS_Mka_00006] [CP_SWS_Mka_00022] [CP_SWS_Mka_00023] [CP_SWS_Mka_CONSTR_00019] [CP_SWS_Mka_CONSTR_00020]
[FO_RS_MACsec_-00030]	Support of Integrity Check Value Key (ICK)	[CP_SWS_Mka_00006] [CP_SWS_Mka_00022]
[FO_RS_MACsec_-00031]	Support of Key Derivation Function (KDF)	[CP_SWS_Mka_00007]
[FO_RS_MACsec_-00032]	List of minimal supported cipher suites	[CP_SWS_Mka_00009] [CP_SWS_Mka_CONSTR_00019] [CP_SWS_Mka_CONSTR_00020]
[FO_RS_MACsec_-00033]	Validation function for ICVs	[CP_SWS_Mka_00001] [CP_SWS_Mka_00006]
[FO_RS_MACsec_-00034]	Generation function for ICVs	[CP_SWS_Mka_00999]
[FO_RS_MACsec_-00035]	Key Handling with combined HSM and MACsec functionality	[CP_SWS_Mka_00023]
[FO_RS_MACsec_-00036]	Interframe gap configuration of Ethernet controller	[CP_SWS_Mka_00999]
[FO_RS_MACsec_-00037]	MACsec participants per link	[CP_SWS_Mka_00015]
[FO_RS_MACsec_-00038]	MKA SC establishment retry phase	[CP_SWS_Mka_00012]
[FO_RS_MACsec_-00039]	MKA rekey conditions	[CP_SWS_Mka_00013] [CP_SWS_Mka_00014]
[SRS_BSW_00323]	All AUTOSAR Basic Software Modules shall check passed API parameters for validity	[CP_SWS_Mka_91035]
[SRS_BSW_00337]	Classification of development errors	[CP_SWS_Mka_00200] [CP_SWS_Mka_00201] [CP_SWS_Mka_00202] [CP_SWS_Mka_00203] [CP_SWS_Mka_91035]
[SRS_BSW_00385]	List possible error notifications	[CP_SWS_Mka_00200] [CP_SWS_Mka_00201] [CP_SWS_Mka_00202] [CP_SWS_Mka_00203] [CP_SWS_Mka_91035]

Table 6.1: RequirementsTracing

7 Functional specification

7.1 Background and rationale

A detailed description of the MACsec and MACsec Key Agreement protocols is included in [3, RS_MACsec] chapter 4.1.

7.2 Motivation

The aim of this document is to specify how to integrate the MKA Module in the Software Layered Architecture of the AUTOSAR Classic Platform.

The purpose of the MACsec Key Agreement Module is to provide a method for discovering MACsec peers and negotiate the security keys needed to secure the link. The MKA Module is responsible for:

- Generating (outgoing) and processing (incoming) MKPDUs.
- Identify and authenticate other partners belonging to the same Connectivity Association (CA).
- Configure and supply the parameters and cryptographic data to the MACsec Entity (per SW or HW) for the respective Secure Channel and Secure Associations established.
- Keep the established Secure Channel (SC) and Secure Association (SA) information updated.
- Refresh keys for an specific Secure Channel to allow exchanging Secure Association Keys (SAKs) without disrupting the communication channel.

The MKA module supports:

- The configuration, initialization, and maintenance of Port Access Entities (PAEs).
- The configuration, initialization, shutdown, and maintenance of MKA Entities (KaYs) which belong to an specific PAE.
- The communication with other AUTOSAR Modules to initialize, update, and shutdown MACsec related parameters into the MACsec Entity (SecY).

The limitations of the referred IEEE standards are included in chapter 4.

Figure 7.1 depicts the MACsec Key Agreement protocol parameter sets exchanged to establish a Secure Channel and finally enable MACsec protected secure communication.

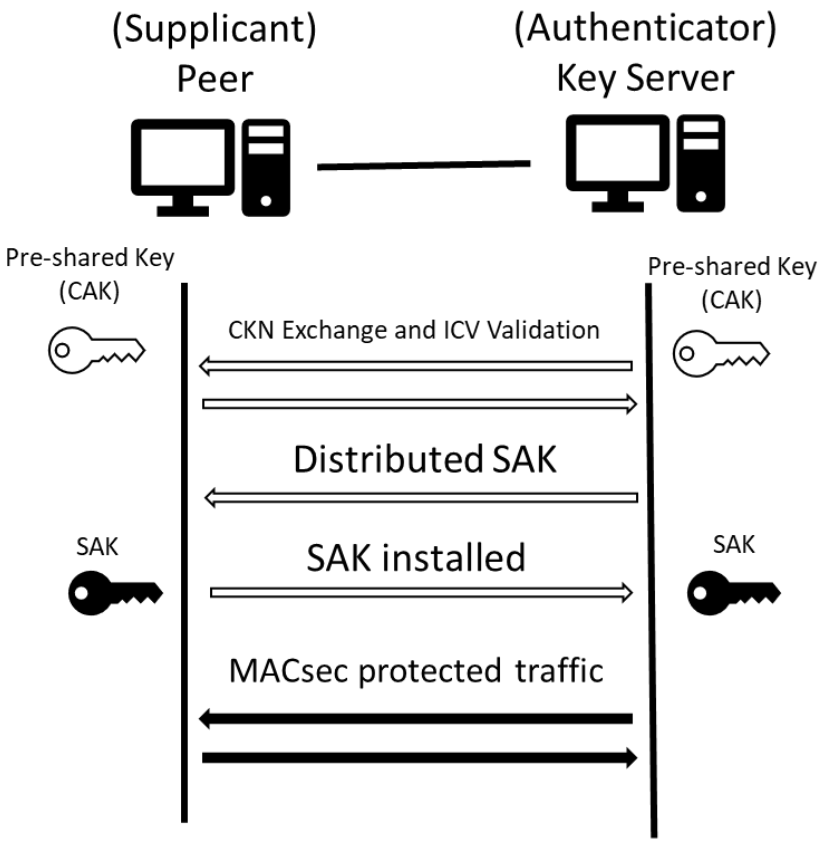

Figure 7.1: fig: MACsec Key Agreement sequence with pre-shared key

Figure 7.2 provides an architecture overview of the AUTOSAR MKA module in the Layered Software Architecture.

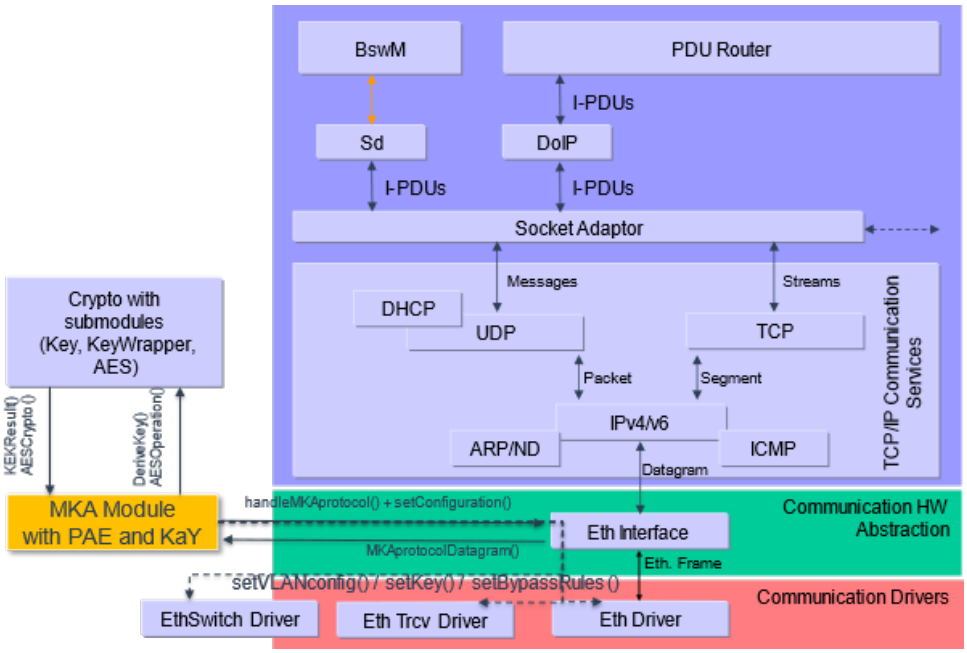

Figure 7.2: MKA module in the SW Architecture of AUTOSAR CP

Based on the IEEE standards ([4, IEEE-802.1AE-2018] and [5, IEEE-802.1X-2020]), the system allows to configure, setup, and run integrity protected and/or encrypted data communication per Ethernet port. This implies that the architecture, specification, and the later implementation enables MACsec on each port when this is configured via AUTOSAR configuration. Additionally, this configuration is not only restricted towards ports. MACsec allows to “bypass” traffic based on EtherType or VLAN-ID. This means that MACsec lets selected traffic pass unprotected.
The usage of MACsec will be statically configured in advance for the entities supporting the protocol, the configuration will be based on rules. This includes rules for determining the bypassed traffic. Since the bypassed traffic can be based on VLAN IDs, the handling of MACsec protected networks interacts with VLAN-based communication. Bypassed traffic is available as soon as a link-up of the transceiver occurs, while protected traffic needs to wait for MACsec and its Key-Agreement sequence to finish first.

The Ethernet Interface will behave different for protected and unprotected traffic. The Ethernet Interface (EthIf) sequence is modified in case the controller (and therefore EthSwt and/or EthTrcv) has to deal with a MACsec protected port.

After receiving a Link-up indication from a MACsec protected port (which could be after a Switch), the Ethernet Interface will trigger the MKA Module to start the MKA Sequence for the corresponding port. Once the MKA module signals the success of the MKA sequence and therefore the proper configuration of MACsec on the port, the Link-Up is propagated to the upper layers (i.e., EthSM or others through the corresponding UL_LinkStateChg method). This is essential to start the upper layer protocols (e.g., SOME/IP-SD) as soon as the MACsec protected link is ready to be used.

This applies as well in case Groups of Ports are defined for the respective network.

7.2.1 Functional components

7.2.1.1 Port Access Entity (PAE)

In the [5, IEEE-802.1X-2020] standard the Port Access Entity (PAE) describes the (SW) entity that controls an Ethernet port. This includes allowing traffic to flow or not to flow but also controlling the MACsec functionality based on the MACsec Key Agreement protocol (MKA).

[5, IEEE-802.1X-2020] defines port based authentication over EAP and, as a particular use case, the MACsec Key Agreement protocol over EAP. In the current version of this document, authentication over EAP is not supported and therefore only authentication based on pre-shared CA Keys (CAKs) is relevant.

The PAE will take care of orchestrating the initialization and shutdown of MKA instances (defined in next section) and setting the status (enabled/disabled) of the physical or virtual controlled port based on the feedback provided by the underlying KaYs.

For a better understanding of the PAE structure, refer to [5, IEEE-802.1X-2020] chapter 12.

7.2.1.2 MACsec Key Agreement Entity (KaY)

Each PAE can have one or multiple MACsec Key Agreement participants (KaY participants) depending on the CKNs assigned to the Port Access Entity.
Each KaY is responsible of recognizing peers which belong to the same CA, distribute and/or install SAKs, and keep the MACsec information up to date during the SC lifetime.
The KaY handles the MKA protocol behavior, including the generation and process of MKPDUs, the control of the cipher suites to use and, the maintenance of the MACsec related parameters of the MACsec Entity (SecY), including Keys (SAKs).

7.3 General Requirements

[CP_SWS_Mka_00001]{DRAFT} ⌈The MKA Module shall implement the EAP-MKA protocol version 3 as specified in [5, IEEE-802.1X-2020] chapter 9 and AUTOSAR Foundation [3, RS_MACsec].⌋*(FO_RS_MACsec_00002, FO_RS_MACsec_00003, FO_RS_MACsec_00023, FO_RS_MACsec_00024, FO_RS_MACsec_00025, FO_RS_MACsec_00029, FO_RS_MACsec_00033)*

Note: The MKA Module should be modeled as described in [5, IEEE-802.1X-2020] chapter 12.
For the excluded parts please refer to section 4.1.

[CP_SWS_Mka_00002]{DRAFT} ⌈The MKA Module shall support 1 to n independent Port Access Entities (PAEs) running at the same time through different ports.⌋*(FO_RS_MACsec_00002, FO_RS_MACsec_00003, FO_RS_MACsec_00004)*

Note: Each physical (Switch port or transceiver) or virtual (MACsec per SW) port will support 0 (No MACsec) or 1 PAE.

[CP_SWS_Mka_00003]{DRAFT} ⌈The MKA Module shall support a list of VLANs to get MACsec bypassed per PAE (i.e. per physical (Switch port or transceiver) or virtual (MACsec per SW) port).
The list of bypassed VLANs shall be provided per configuration (`MkaBypassVlan`).⌋*(FO_RS_MACsec_00002, FO_RS_MACsec_00007, FO_RS_MACsec_00008)*

Note: The MACsec by-passed traffic will be unprotected traffic through the port.

[CP_SWS_Mka_00004]{DRAFT} ⌈The MKA Module shall support a list of EtherTypes to get MACsec bypassed per PAE (i.e. per physical (Switch port or transceiver) or virtual (MACsec per SW) port).
The list of bypassed EtherTypes shall be provided per configuration (`MkaBypas-`

sEtherType).⌋(*FO_RS_MACsec_00002*, *FO_RS_MACsec_00007*, *FO_RS_MACsec_00008*)

Note: The MACsec bypassed traffic will be unprotected traffic through the port.

[CP_SWS_Mka_00005]{DRAFT} ⌈The MKA Module shall support configuring 1 to n CKNs in an specific Port Access Entity (PAE).
Each configured CKN will start a parallel MACsec participant entity (i.e. MkaKayParticipant) through the mentioned PAE.
Repeated CKNs shall be treated as one for an specific PAE.⌋(*FO_RS_MACsec_00002*, *FO_RS_MACsec_00024*, *FO_RS_MACsec_00025*)

Note: It is recommended to implement a configuration check to avoid duplicated CKNs referred under the same MkaKay instance.

[CP_SWS_Mka_00006]{DRAFT} ⌈An MKA KaY participant (MkaKayParticipant) shall not start transmitting or processing MKPDUs until its respective CAK is available and the derived keys (ICK and KEK) are ready.⌋(*FO_RS_MACsec_00002*, *FO_RS_MACsec_00003*, *FO_RS_MACsec_00025*, *FO_RS_MACsec_00029*, *FO_RS_MACsec_00030*, *FO_RS_MACsec_00033*)

[CP_SWS_Mka_00007]{DRAFT} ⌈The MKA Module shall support generation of SAKs based on:

- Key Derivation Function (KDF), see [5, IEEE-802.1X-2020] chapter 9.8.1.
- Random Number Generator (RNG)

⌋(*FO_RS_MACsec_00002*, *FO_RS_MACsec_00031*)

[CP_SWS_Mka_00008]{DRAFT} ⌈The MKA Module shall support the following MKPDU Parameter sets:

- Basic Parameter Set
- Live Peer List → Parameter set type 1
- Potential Peer List → Parameter set type 2
- MACsec SAK Use → Parameter set type 3
- Distributed SAK → Parameter set type 4
- Announcement → Parameter set type 7
- XPN → Parameter set type 8

⌋(*FO_RS_MACsec_00002*, *FO_RS_MACsec_00003*, *FO_RS_MACsec_00010*, *FO_RS_MACsec_00017*, *FO_RS_MACsec_00023*)

[CP_SWS_Mka_00009]{DRAFT} ⌈The MKA Module shall implement the EAPoL-MKA-Announcement with TLV type 112 (MACsec cipher suites) as specified in [5, IEEE-

802.1X-2020] chapter 11.12.3.⌋*(FO_RS_MACsec_00002, FO_RS_MACsec_00027, FO_RS_MACsec_00028, FO_RS_MACsec_00032)*

Note: The MACsec cipher suite announcement serves for the Key Server to recognize the ciphers supported by the other end.
The EAPoL-Announcement TLV shall be transmitted as a parameter on an EAPoL-MKA Announcement Parameter Set as defined in Figure 11-15 of [5, IEEE-802.1X-2020].

[CP_SWS_Mka_00011]{DRAFT} ⌈The role of an MKA instance (`MkaKay`) shall be set per configuration (i.e. `MKA_KEY_SERVER` or `MKA_PEER`) (`MkaRole`).
The Key Server priority shall be configurable (`MkaKeyServerPriority`), in case it is not specifically provided in configuration the following values shall be used:

- Key Server = 0
- Peer = 255

⌋*(FO_RS_MACsec_00002, FO_RS_MACsec_00003)*

[CP_SWS_Mka_00012]{DRAFT} ⌈The MKA Module shall support retry for the MKA sequence.
If an MKA KaY participant (`MkaKayParticipant`) cannot successfully identify or successfully establish a SC with any participant in the link , it should retry the MKA sequence following a per configuration parametrized *retry base delay with Exponential Back-off* (`MkaRetryBaseDelay`) until a *retry cyclic delay* (`MkaRetryCyclicDelay`).⌋*(FO_RS_MACsec_00038)*

Note: As an example, in case `MkaRetryBaseDelay` = 0.02 and `MkaRetryCyclicDelay` = 0.5, the retry sequence will be as follows 20ms, 40ms, 80ms, 160ms, 320ms, 500ms, 500ms, ...

[CP_SWS_Mka_00013]{DRAFT} ⌈The MKA Instances (`MkaKay`) configured with the Key Server (`MKA_KEY_SERVER`) role shall support re-keying distributed SAKs after a configurable time span (`MkaSakRekeyTimeSpan`).⌋*(FO_RS_MACsec_00039)*

Note: The time span is set per configuration.

[CP_SWS_Mka_00014]{DRAFT} ⌈The MKA Instances configured with the Key Server (`MKA_KEY_SERVER`) role shall rekey distributed SAKs in case the packet number space of one direction (sending or receiving) exceeds:

- 0xC000 0000 for 32-bit PNs.
- 0xC000 0000 0000 0000 for 64-bit PNs (XPN mode).

⌋*(FO_RS_MACsec_00039)*

Note: This is required in the [5, IEEE-802.1X-2020] standard, chapter 9.8.

[CP_SWS_Mka_00015]{DRAFT} ⌈Each MKA instance shall assume exactly two participants per link. Therefore, having exactly one peer.⌋*(FO_RS_MACsec_00002, FO_RS_MACsec_00037)*

Note: This implies, that one must take the Key Server role and the other the peer role. This requirement permits a MKA instance to immediately continue with the MKA sequence after detecting and successfully authenticating another participant in the link which belongs to the same CA, avoiding start-up delays.

7.4 Limitations

An overview of non-supported features can be found in chapter 4.

7.4.1 Limitations on MKA Entity

[CP_SWS_Mka_00016]{DRAFT} ⌈The MKA Module may support authentication based on EAP as detailed in [5, IEEE-802.1X-2020] chapter 8.⌋*(FO_RS_MACsec_00002, FO_RS_MACsec_00024)*

Note: Authentication based on EAP is not required as the mutual authentication of participants is achieved based on pre-shared Keys.

[CP_SWS_Mka_00017]{DRAFT} ⌈The MKA Module shall support only one MKA Participant (`MkaKayParticipant`) to success per port (i.e. per PAE `MkaPaeInstance`). If one KaY participant is able to correctly establish a SC, the other started participants (`MkaKayParticipant`) of the same PAE shall give up.⌋*(FO_RS_MACsec_00002)*

Note: As specified in [CP_SWS_Mka_00005], a PAE instance can initiate 1 to n MKA participant instances (`MkaKayParticipant`) but only one of them shall succeed configuring a Secure Channel in the port (Point-to-Multipoint architecture is not supported).

7.5 Cryptographic requirements

[CP_SWS_Mka_CONSTR_00019]{DRAFT} ⌈The MKA Module shall support the following Cipher suites to be configured in the MACsec Entity (either per SW or HW):

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

⌋(*FO_RS_MACsec_00002*, *FO_RS_MACsec_00020*, *FO_RS_MACsec_00026*, *FO_RS_MACsec_00027*, *FO_RS_MACsec_00028*, *FO_RS_MACsec_00029*, *FO_RS_MACsec_00032*)

Note: The MKA Module shall support announcing and configuring the listed ciphers (`MkaMacSecCipherSuite`).
Detailed information can be found in [5, IEEE-802.1X-2020] Figure 11-12 and [4, IEEE-802.1AE-2018] chapter 14.3.

[CP_SWS_Mka_CONSTR_00020]{DRAFT} ⌈The MKA Module shall support configuring 1 to 4 cipher suites per `MkaCryptoAlgoConfig`, each of them with an unique `MkaMacSecCipherSuitePrio`.
The `MkaMacSecCipherSuitePrio` shall accept the value 1 to 4, being 1 the higher priority and 4 the lowest priority.⌋(*FO_RS_MACsec_00002*, *FO_RS_MACsec_00020*, *FO_RS_MACsec_00026*, *FO_RS_MACsec_00027*, *FO_RS_MACsec_00028*, *FO_RS_MACsec_00029*, *FO_RS_MACsec_00032*)

Note: The `MkaMacSecCipherSuitePrio` parameter shall be used by a `MkaKayParticipant` with `MkaRole` = `MKA_KEY_SERVER` to select the cipher suite to use for MACsec with the other participant within the common cipher suites supported (shared with the EAPoL-MKA-Announcement “MACsec cipher suites”).

The cryptographic operations like the derivation of MACsec keys and authentication based on CAK pre-shared keys should be delegated to the CSM Module.

Note: For detailed information, refer to [5, IEEE-802.1X-2020] chapter 9.3.

[CP_SWS_Mka_00022]{DRAFT} ⌈Derived keys (specifically ICKs and KEKs) may get pre-calculated and stored in tamper proof manner to optimize the initialization time of the module.
SAKs are implicitly excluded from this requirement. SAKs must not be pre-calculated neither reused.⌋(*FO_RS_MACsec_00029*, *FO_RS_MACsec_00030*)

[CP_SWS_Mka_00023]{DRAFT} ⌈It shall be supported that Secure Association Keys (SAKs) can directly be installed from a HSM to a MACsec Entity (SecY).⌋(*FO_RS_MACsec_00029*, *FO_RS_MACsec_00035*)

7.6 Communication with MACsec Entity (SecY)

The MKA Module, and particularly a specific MKA Entity (KaY) running over a PAE, shall initialize and maintain a Secure Channel for MACsec over an specific MACsec Entity (SecY). The mentioned MACsec Entity can be a SW implementation of MACsec in lower layers or a HW implementation in a Phy or Switch.

This shall be achieved with the API specified in chapter 8.

[CP_SWS_Mka_00024]{DRAFT} ⌈The MKA Module shall propagate the MACsec Entity specific parameters as needed by means of the EthIf API.
This requirement applies for the initialization, shutdown and, maintenance of MACsec related parameters.⌋*(FO_RS_MACsec_00002, FO_RS_MACsec_00003)*

[CP_SWS_Mka_00025]{DRAFT} ⌈The MKA Module shall collect the MACsec Entity (SecY) statistics when requested by means of the EthIf API.⌋*(FO_RS_MACsec_00009)*

Note: Other modules may request port specific MACsec statistics in order to set DTCs, answer to Diagnostics, and for monitoring.

7.7 Configurable behavior of MKA

[CP_SWS_Mka_00026]{DRAFT} ⌈The status and behavior of a specific `MkaPaeInstance` shall be configurable per initial configuration.⌋*(FO_RS_MACsec_00005)*

Note: In case the proposal from [5, IEEE-802.1X-2020] chapter 12.5 is used, the variable *useEAP* is currently not supported (that means, the value shall be per default set to *Never*).

[CP_SWS_Mka_00027]{DRAFT} ⌈It shall be possible to set the configuration of a specific PAE dynamically through the MKA module API.
The change shall apply in the next power cycle.
If a configuration parameter (e.g. through `Mka_SetPaePermissiveMode`, `Mka_SetCknStatus`, or `Mka_SetEnable`) is changed by means of the API, the original per configuration set value shall be ignored and the in NVRAM persisted value shall be used from the next power cycle onwards.⌋*(FO_RS_MACsec_00005)*

[CP_SWS_Mka_00028]{DRAFT} ⌈In case `MkaOnFailPermissiveMode` == `TIMEOUT` and `MkaParticipantActivate` == TRUE for a specific `MkaKayParticipant`, it shall determine that the MKA has failed when all these conditions are met:

- MKA sequence did not succeed → The participants could not reach the state in which the SAK is installed for Rx and Tx.
- MKA timeout (`MkaOnFailPermissiveModeTimeout`) is exceeded.

If all these conditions are met, the error MKA_E_TIMEOUT shall be triggered.⌋*(FO_RS_MACsec_00005)*

Note: The MKA timeout value is set per configuration with the `MkaOnFailPermissiveModeTimeout` parameter.

[CP_SWS_Mka_00029]{DRAFT} ⌈The MKA timer for `MkaOnFailPermissiveModeTimeout` shall start counting after LinkUp (*ETH_MODE_ACTIVE*) of the referred physical or virtual port.⌋*(FO_RS_MACsec_00005)*

[CP_SWS_Mka_00030]{DRAFT} ⌈The `MkaOnFailPermissiveModeTimeout` value shall be reset if any of the following conditions is met:

- After start up.
- On the transition from LinkDown (*ETH_MODE_DOWN*) to LinkUp (*ETH_MODE_ACTIVE*) of the referred physical or virtual port.

⌋(*FO_RS_MACsec_00005*)

7.7.1 MKA behavior

[CP_SWS_Mka_00031]{DRAFT} ⌈A MKA Entity (KaY) shall start the MKA sequence(s) through the referred EthIf (`MkaEthIfControllerRef`) immediately after receiving the port link-up signal with the `Mka_LinkStateChange` function.⌋(*FO_RS_MACsec_00002*)

[CP_SWS_Mka_00032]{DRAFT} ⌈A MKA Entity (KaY) shall signal the successful configuration of the MACsec protected port to the EthIf with the `EthIf_MacSecOperational` or `EthIf_SwitchMacSecOperational` function.⌋(*FO_RS_MACsec_00002*)

Note: A MKA Entity determines that the MACsec protected port is successfully configured as soon as MACsec protected frames can be transmitted and received from both participants.

7.7.2 MKA Error Handling

To ease complexity of the implementation, the MKA module may heal certain extended production errors automatically at start-up.

[CP_SWS_Mka_00033]{DRAFT} ⌈If one or more CAKs referenced by `MkaCryptoCknCakKeyRef` is/are not initialized, MKA_E_KEY_NOT_PRESENT_INSTANCE shall be set to "Fail".⌋()

Note: Also see CSM Return Code CRYPTO_E_KEY_NOT_AVAILABLE.

[CP_SWS_Mka_00034]{DRAFT} ⌈If the verification of the ICV of MKPDUs or the unwrapping of keys fails for one or more CKNs because of a wrong key, MKA_E_KEY_MISMATCH_INSTANCE shall be set to "Fail".⌋()

[CP_SWS_Mka_00035]{DRAFT} ⌈If the MKA peer only supports incompatible cipher suites, MKA_E_ALGO_MISMATCH_INSTANCE shall be set to "Fail".⌋()

[CP_SWS_Mka_00036]{DRAFT} ⌈MKA_E_TIMEOUT_INSTANCE, MKA_E_KEY_NOT_PRESENT_INSTANCE, MKA_E_KEY_MISMATCH_INSTANCE, and MKA_E_ALGO_MISMATCH_INSTANCE shall be healed (set to "Pass"), when the error condition is not applicable anymore.⌋()

Note: If an implementer chooses to not implement [CP_SWS_Mka_00036], the mentioned errors shall be healed on start-up of the MKA module.

7.8 Error Classification

Section "Error Handling" of the document [2] "General Specification of Basic Software Modules" describes the error handling of the Basic Software in detail. Above all, it constitutes a classification scheme consisting of five error types which may occur in BSW modules.

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

7.8.1 Development Errors

[CP_SWS_Mka_91035]{DRAFT} ⌈

Type of error	*Related error code*	*Error value*
MKA Component initiated with null configuration	MKA_E_CFG_NULL_PTR	0x01
API called with invalid parameter value.	MKA_E_INVALID_PARAMETER	0x04
API called with a NULL pointer.	MKA_E_PARAM_POINTER	0x05
API called prior module is initialized.	MKA_E_UNINIT	0x06

⌋(*SRS_BSW_00337*, *SRS_BSW_00385*, *SRS_BSW_00323*)

7.8.2 Runtime Errors

There are no runtime errors.

7.8.3 Transient Faults

There are no transient faults.

7.8.4 Production Errors

There are no production errors.

7.8.5 Extended Production Errors

7.8.5.1 MKA_E_TIMEOUT_INSTANCE

[CP_SWS_Mka_00200] ⌈

Error Name:	MKA_E_TIMEOUT_INSTANCE	
Short Description:	MKA timeout while negotiating with remote peer.	
Long Description:	MKA timeout while negotiating with remote peer. In case `MkaOnFailPermissiveMode` == `TIMEOUT` and `MkaOnFailPermissiveModeTimeout` is overflowed, the error will be set.	
Detection Criteria:	Fail	If the PAE instance's `MkaOnFailPermissiveMode` == `TIMEOUT` and `MkaOnFailPermissiveModeTimeout` is overflowed, the error will be set.
	Pass	If the PAE instance's `MkaOnFailPermissiveMode` == `NEVER` or If the PAE instance's `MkaOnFailPermissiveMode` == `TIMEOUT`, and the MkaKay could establish a transmission and reception SC with a peer before `MkaOnFailPermissiveModeTimeout` is reached, the error is not set.
Secondary Parameters:	Not Applicable	
Time Required:	The time to detect the error is linked to the `MkaOnFailPermissiveMode` and `MkaOnFailPermissiveModeTimeout`.	
Monitor Frequency:	Once after port's link-up per port.	
MIL illumination:	Not Applicable	

⌋*(SRS_BSW_00385, SRS_BSW_00337)*

7.8.5.2 MKA_E_KEY_NOT_PRESENT_INSTANCE

[CP_SWS_Mka_00201] ⌈

Error Name:	MKA_E_KEY_NOT_PRESENT_INSTANCE	
Short Description:	Necessary keys not present to initiate MKA negotiation.	
Long Description:	Pre-shared keys (i.e. CAK, ICK or KEK) to start the MKA sequence are not present in at least one of the active configured Kay Participants.	
Detection Criteria:	Fail	The pre-shared keys of an active `MkaKayParticipant` are not present.
	Pass	All Kay participants have the needed pre-shared keys present.
Secondary Parameters:	Not Applicable	
Time Required:	0.5	
Monitor Frequency:	once-per-trip	
MIL illumination:	Not Applicable.	

⌋(*SRS_BSW_00385*, *SRS_BSW_00337*)

7.8.5.3 MKA_E_KEY_MISMATCH_INSTANCE

[CP_SWS_Mka_00202] ⌈

Error Name:	MKA_E_KEY_MISMATCH_INSTANCE	
Short Description:	MKA negotiation failed due to key mismatch with remote peer (MKPDUs ICV mismatch).	
Long Description:	MKA negotiation failed due to key mismatch with remote peer (MKPDUs ICV mismatch). Triggered in case MKPDU cannot be validated for received MKPDUs which distribute a matching CKN.	
Detection Criteria:	Fail	A received MKPDU with matching CKN cannot be successfully validated.
	Pass	All received MKPDUs with matching CKN are successfully validated.
Secondary Parameters:	Not Applicable	
Time Required:	Not Applicable.	
Monitor Frequency:	Continuous	
MIL illumination:	Not Applicable.	

⌋(*SRS_BSW_00385*, *SRS_BSW_00337*)

7.8.5.4 MKA_E_ALGO_MISMATCH_INSTANCE

[CP_SWS_Mka_00203] ⌈

Error Name:	MKA_E_ALGO_MISMATCH_INSTANCE	
Short Description:	MKA negotiation failed due to incompatible cipher suite with remote peer.	
Long Description:	MKA Negotiation failed due to incompatible cipher suite with remote peer. Triggered in case the participants in the MKA communication do not support any MACsec cipher suite in common and therefore cannot distribute neither install a valid SAK.	
Detection Criteria:	Fail	The KaY participants of a communication (local and remote) do not support a common MACsec cipher suite.
	Pass	The KaY participants of a communication (local and remote) support one or more common MACsec cipher suites.
Secondary Parameters:	Not Applicable	
Time Required:	Not Applicable.	
Monitor Frequency:	Continuous	
MIL illumination:	Not Applicable.	

⌋*(SRS_BSW_00385, SRS_BSW_00337)*

8 API specification

8.1 Imported types

In this chapter all types included from the following files are listed.

[CP_SWS_Mka_91005]{DRAFT} ⌈

Module	*Header File*	*Imported Type*
ComStack_Types	ComStack_Types.h	BufReq_ReturnType
Eth	Eth_GeneralTypes.h	Eth_BufIdxType
	Eth_GeneralTypes.h	Eth_FrameType
EthSwt	Eth_GeneralTypes.h	EthSwt_MgmtInfoType
EthTrcv	Eth_GeneralTypes.h	EthTrcv_LinkStateType
IdsM	IdsM_Types.h	IdsM_SecurityEventIdType
NvM	Rte_NvM_Type.h	NvM_BlockIdType
Std	Std_Types.h	Std_ReturnType
	Std_Types.h	Std_VersionInfoType

⌋()

8.2 Type definitions

8.2.1 Mka_MacSecConfigType

[CP_SWS_Mka_91002]{DRAFT} ⌈

Name	Mka_MacSecConfigType (DRAFT)	
Kind	Structure	
Elements	ProtectFrames	
	Type	boolean
	Comment	Indicates status if the MACsec protection of the frames is active or not
	ReplayProtect	
	Type	boolean
	Comment	Indicates status if replay protection is enable or disable
	ReplayWindow	
	Type	uint32
	Comment	If ReplayProtect is enable, indicates the used replay protect window
	ValidateFrames	
	Type	Mka_ValidateFramesType
	Comment	Status of the validation of the frames. See Mka_ValidateFramesType for possible values
	CurrentCipherSuite	
	Type	uint64

▽

△

	Comment	Indicates which cipher suite is used in the SecY to update.
	ConfidentialityOffset	
	Type	Mka_ConfidentialityOffsetType
	Comment	Set the Confidentiality Offset. See Mka_ConfidentialityOffsetType for possible values
	ControlledPortEnabled	
	Type	boolean
	Comment	Status if the controlled port is enabled or disabled
	BypassedVlanPtrs	
	Type	const uint16*
	Comment	Pointer to the list of bypassed VLANs
	BypassedVlansLength	
	Type	uint8
	Comment	Length of the list of bypassed VLANs
	BypassedEtherTypesPtr	
	Type	const uint16*
	Comment	Pointer to the list of the bypassed Ethernet Types
	BypassedEtherTypesLength	
	Type	uint8
	Comment	Length of the list of the bypassed Ethernet Types
Description	Structure to configure a referred SecY **Tags:** atp.Status=DRAFT	
Available via	Mka.h	

⌋()

8.2.2 Mka_ValidateFramesType

[CP_SWS_Mka_91004]{DRAFT} ⌈

Name	Mka_ValidateFramesType (DRAFT)		
Kind	Enumeration		
Range	MKA_VALIDATE_ DISABLED	0	Disable validation, remove SecTAGs and ICVs (if present) from received frames.
	MKA_VALIDATE_ CHECKED	1	Enable validation, do not discard invalid frames
	MKA_VALIDATE_STRICT	2	Enable validation and discard invalid frames
Description	Controls validation of received frames **Tags:** atp.Status=DRAFT		
Available via	Mka.h		

⌋()

8.2.3 Mka_ConfidentialityOffsetType

[CP_SWS_Mka_91003]{DRAFT} ⌈

Name	Mka_ConfidentialityOffsetType (DRAFT)		
Kind	Enumeration		
Range	MKA_CONFIDENTIALITY_NONE	0	Confidentiality protection disabled
	MKA_CONFIDENTIALITY_OFFSET_0	1	Zero initial octets of each user data without confidentiality protection
	MKA_CONFIDENTIALITY_OFFSET_30	2	30 initial octets of each user data without confidentiality protection
	MKA_CONFIDENTIALITY_OFFSET_50	3	50 initial octets of each user data without confidentiality protection
Description	Indicates the offset in case of integrity with confidentiality **Tags:** atp.Status=DRAFT		
Available via	Mka.h		

⌋()

8.2.4 Mka_Stats_Tx_SecYType

[CP_SWS_Mka_91008]{DRAFT} ⌈

Name	Mka_Stats_Tx_SecYType (DRAFT)	
Kind	Structure	
Elements	OutPkts_Untagged	
	Type	uint64
	Comment	The number of packets transmitted without a SecTAG
	OutPkts_TooLong	
	Type	uint64
	Comment	The number of transmitted packets discarded because their length is greater than the accepted maximum length (mtu) of the Port
	OutOctets_Protected	
	Type	uint64
	Comment	The number of plain text octets integrity protected but not encrypted in transmitted frames
	OutOctets_Encrypted	
	Type	uint64
	Comment	The number of plain text octets integrity protected and encrypted in transmitted frames
Description	MACsec Entity (SecY) transmission statistics **Tags:** atp.Status=DRAFT	
Available via	Mka.h	

⌋()

8.2.5 Mka_Stats_Rx_SecYType

[CP_SWS_Mka_91010]{DRAFT} ⌈

Name	Mka_Stats_Rx_SecYType (DRAFT)	
Kind	Structure	
Elements	InPkts_Untagged	
	Type	uint64
	Comment	The number of packets without the MACsec tag (SecTAG) received if Mka_ValidateFrames was not 'MKA_VALIDATE_STRICT'
	InPkts_NoTag	
	Type	uint64
	Comment	The number of received packets without a SecTAG discarded because Mka_ValidateFrames was 'MKA_VALIDATE_STRICT'
	InPkts_BadTag	
	Type	uint64
	Comment	The number of received packets discarded with an invalid SecTAG, zero value PN, or invalid ICV
	InPkts_NoSa	
	Type	uint64
	Comment	The number of received packets with an unknown SCI or for an unused SA by the security entity
	InPkts_NoSaError	
	Type	uint64
	Comment	The number of packets discarded because the received SCI is unknown or the SA is not in use
	InPkts_Overrun	
	Type	uint64
	Comment	The number of packets discarded because they exceeded cryptographic performance capabilities
	InOctets_Validated	
	Type	uint64
	Comment	The number of plaintext octets recovered from packets that were integrity protected but not encrypted
	InOctets_Decrypted	
	Type	uint64
	Comment	The number of plaintext octets recovered from packets that were integrity protected and encrypted
Description	MACsec Entity (SecY) reception statistics **Tags:** atp.Status=DRAFT	
Available via	Mka.h	

⌋()

8.2.6 Mka_Stats_Tx_ScType

[CP_SWS_Mka_91009]{DRAFT} ⌈

Name	Mka_Stats_Tx_ScType (DRAFT)	
Kind	Structure	
Elements	OutPkts_Protected	
	Type	uint64
	Comment	The number of integrity protected but not encrypted packets for this transmit SC
	OutPkts_Encrypted	
	Type	uint64
	Comment	The number of integrity protected and encrypted packets for this transmit SC
Description	Secure Channel transmission statistics **Tags:** atp.Status=DRAFT	
Available via	Mka.h	

⌋()

8.2.7 Mka_Stats_Rx_ScType

[CP_SWS_Mka_91011]{DRAFT} ⌈

Name	Mka_Stats_Rx_ScType (DRAFT)	
Kind	Structure	
Elements	InPkts_Ok	
	Type	uint64
	Comment	The number of packets received for this secure channel successfully validated and within the replay window
	InPkts_Unchecked	
	Type	uint64
	Comment	The number of packets received for this secure channel, if Mka_ ValidateFrames was 'MKA_VALIDATE_DISABLED'
	InPkts_Delayed	
	Type	uint64
	Comment	The number of received packets, for this secure channel, with packet number (PN) lower than the lowest acceptable packet number (Lowest Pn) and ReplayProtect was false
	InPkts_Late	
	Type	uint64
	Comment	The number of packets discarded, for this secure channel, because the received packet number (PN) was lower than the lowest acceptable packet number (LowestPn) and ReplayProtect was true
	InPkts_Invalid	
	Type	uint64

▽

△

	Comment	The number of packets, for this secure channel, that failed validation but could be received because ValidateFrames was 'MKA_VALIDATE_CHECKED' and the data was not encrypted (so the original frame could be recovered)
	InPkts_NotValid	
	Type	uint64
	Comment	The number of packets discarded, for this secure channel, because validation failed and ValidateFrames was 'MKA_VALIDATE_STRICT' or the data was encrypted (so the original frame could not be recovered)
Description	Secure Channel reception statistics **Tags:** atp.Status=DRAFT	
Available via	Mka.h	

⌋()

8.2.8 Mka_SakKeyPtrType

[CP_SWS_Mka_91013]{DRAFT} ⌈

Name	Mka_SakKeyPtrType (DRAFT)	
Kind	Structure	
Elements	HashKeyPtr	
	Type	const uint8*
	Comment	Pointer to the Hash Key
	SakKeyPtr	
	Type	const uint8*
	Comment	Pointer to the SAK
	SaltKeyPtr	
	Type	const uint8*
	Comment	Pointer to the Salt
Description	SAK key reference **Tags:** atp.Status=DRAFT	
Available via	Mka.h	

⌋()

8.2.9 Mka_PermissiveModeType

[CP_SWS_Mka_91012]{DRAFT} ⌈

Name	Mka_PermisiveModeType (DRAFT)		
Kind	Enumeration		
Range	NEVER	0	The controlled port will never be set to enabled if the participants cannot establish and successfully use a MACsec Secure Channel.
	TIMEOUT	1	The controlled port will be set to enabled and MACsec will not be used in the referred port if the timeout value (MkaOnFailPermissive Mode Timeout) is reached and none MKA instance under the PAE instance could success the following conditions: - A participant belonging to the same CA was recognized and authenticated. - A secure channel could be established. - Both participants can transmit and receive MACsec protected traffic through the SC.
Description	Permissive modes of MKA **Tags:** atp.Status=DRAFT		
Available via	Mka.h		

⌋()

8.2.10 Mka_Stats_SecYType

[CP_SWS_Mka_91028]{DRAFT} ⌈

Name	Mka_Stats_SecYType (DRAFT)	
Kind	Structure	
Elements	StatsTxPhy	
	Type	Mka_Stats_Tx_SecYType
	Comment	Set of statistics in the Security Entity Phy by transmision
	StatsRxPhy	
	Type	Mka_Stats_Rx_SecYType
	Comment	Set of statistics in the Security Entity Phy by reception
	StatsTxSc	
	Type	Mka_Stats_Tx_ScType
	Comment	Set of statistics in the Security Entity's Secure Channel by reception
	StatsRxSc	
	Type	Mka_Stats_Rx_ScType
	Comment	Set of statistics in the Security Entity's Secure Channel by reception
Description	Security Entity statistics **Tags:** atp.Status=DRAFT	
Available via	Mka.h	

⌋()

8.2.11 Mka_PaeStatusType

[CP_SWS_Mka_91027]{DRAFT} ⌈

Name	Mka_PaeStatusType (DRAFT)	
Kind	Structure	
Elements	ConnectionStatus	
	Type	Mka_MkaStatus
	Comment	Status of the MKA
	PeerSci	
	Type	uint64
	Comment	SCI includes the peer's MAC and port
	CknInUse	
	Type	Array of unsigned char[32]
	Size	32
	Comment	CKN used for the establishment of the MACsec Secure Channel
Description	Structure with the specific information of a PAE **Tags:** atp.Status=DRAFT	
Available via	Mka.h	

⌋()

8.2.12 Mka_MkaStatusType

[CP_SWS_Mka_91025]{DRAFT} ⌈

Name	Mka_MkaStatus (DRAFT)		
Kind	Enumeration		
Range	MKA_STATUS_MACSEC_RUNNING	0	MKA session key has been agreed and MACsec link is currently up
	MKA_STATUS_WAITING_PEER_LINK	1	MKA is waiting for a link up of the underlying device to begin negotiation
	MKA_STATUS_WAITING_PEER	2	MKA is waiting for a remote peer to transmit MKPDU's to begin negotiation
	MKA_STATUS_IN_PROGRESS	3	MKA negotiation is ongoing
	MKA_STATUS_AUTH_FAIL_UNKNOWN_PEER	4	MKA negotiation is not possible because ICV's of remote peer are invalid (ICK and therefore CAK keys are different)
	MKA_STATUS_UNDEFINED	0xFF	Undefined state, reported when the given bus is disabled
Description	Status of the MKA instance. **Tags:** atp.Status=DRAFT		
Available via	Mka.h		

⌋()

8.2.13 Mka_ConfigType

[CP_SWS_Mka_91026]{DRAFT} ⌈

Name	Mka_ConfigType (DRAFT)
Kind	Structure
Description	Implementation specific structure of the post build configuration **Tags:** atp.Status=DRAFT
Available via	Mka.h

⌋()

8.3 Function definitions

8.3.1 Mka_Init

[CP_SWS_Mka_91001]{DRAFT} ⌈

Service Name	Mka_Init (DRAFT)	

Syntax	`Std_ReturnType Mka_Init (`
	`  const Mka_ConfigType* ConfigPtr`
	`)`	

Service ID [hex]	0x1	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	ConfigPtr	Points to the implementation specific structure
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	Initializes the MKA module **Tags:** atp.Status=DRAFT	
Available via	Mka.h	

⌋()

8.3.2 Mka_GetVersionInfo

[CP_SWS_Mka_91014]{DRAFT} ⌈

Service Name	Mka_GetVersionInfo (DRAFT)

Syntax	`Std_ReturnType Mka_GetVersionInfo (`
	`  Std_VersionInfoType* VersionInfoPtr`
	`)`

Service ID [hex]	0x2

▽

△

Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	None	
Parameters (inout)	None	
Parameters (out)	VersionInfoPtr	Version information of this module
Return value	Std_ReturnType	E_OK: The request has been accepted E_NOT_OK: The request has not been accepted
Description	Returns the version information of this module **Tags:** atp.Status=DRAFT	
Available via	Mka.h	

⌋()

8.3.3 Mka_SetCknStatus

[CP_SWS_Mka_91015]{DRAFT} ⌈

Service Name	Mka_SetCknStatus (DRAFT)	

Syntax	`Std_ReturnType Mka_SetCknStatus (`
	`  uint8 MkaPaeIdx,`
	`  boolean Enable,`
	`  const uint8* Ckn,`
	`  uint8 CknLength`
	`)`	

Service ID [hex]	0x3	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different MkaPaeIdx, Non reentrant for the same MkaPaeIdx	
Parameters (in)	MkaPaeIdx	Index of the PAE within the context of the MKA module
	Enable	Boolean to control the Mka Participant Activate status. True -> The MKA Participant exchanges MKPDUs. False -> The MKA Participant does not exchange MKPDUs.
	Ckn	Connectivity Association Key Name to identify the KaY participant
	CknLength	Length of the CKN parameter provided
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	Set status of a CKN from a PAE **Tags:** atp.Status=DRAFT	
Available via	Mka.h	

⌋()

[CP_SWS_Mka_01001]{DRAFT} ⌈The function `Mka_SetCknStatus` shall set the activation status of the `MkaKayParticipant` which contains the provided `Ckn` under the provided `MkaPaeIdx`.
The new activation status shall be persistently stored in NVM and used from next power cycle onwards (as required in [CP_SWS_Mka_00030]).

The per configuration provided activation status (MkaParticipantActivate) of the MkaKayParticipant shall not be used if a valid value is stored on the NVM.⌋()

8.3.4 Mka_GetCknStatus

[CP_SWS_Mka_91016]{DRAFT} ⌈

Service Name	Mka_GetCknStatus (DRAFT)	
Syntax	`Std_ReturnType Mka_GetCknStatus ( uint8 MkaPaeIdx, const uint8* Ckn, uint8 CknLength, boolean* EnablePtr )`	
Service ID [hex]	0x4	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different MkaPaeIdx, Non reentrant for the same MkaPaeIdx	
Parameters (in)	MkaPaeIdx	Index of the PAE within the context of the MKA module
	Ckn	Connectivity Association Key Name to identify the KaY participant
	CknLength	Length of the CKN parameter provided
Parameters (inout)	None	
Parameters (out)	EnablePtr	Pointer to the Mka Participant activation status
Return value	Std_ReturnType	E_OK: The request has been accepted E_NOT_OK: The request has not been accepted
Description	Get Status of a CKN from a PAE **Tags:** atp.Status=DRAFT	
Available via	Mka.h	

⌋()

8.3.5 Mka_SetEnable

[CP_SWS_Mka_91020]{DRAFT} ⌈

Service Name	Mka_SetEnable (DRAFT)	
Syntax	`Std_ReturnType Mka_SetEnable ( uint8 MkaPaeIdx, boolean Enable )`	
Service ID [hex]	0x8	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different MkaPaeIdx, Non reentrant for the same MkaPaeIdx	
Parameters (in)	MkaPaeIdx	Index of the PAE within the context of the MKA module
	Enable	Boolean to control the Mka activation of a 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	Set the MKA activation status of a PAE **Tags:** atp.Status=DRAFT	
Available via	Mka.h	

⌋()

[CP_SWS_Mka_01002]{DRAFT} ⌈The function `Mka_SetEnable` shall set the activation status of the `MkaKay` of the provided `MkaPaeIdx`.
The new activation status shall be persistently stored in NVM and used from next power cycle onwards (as required in [CP_SWS_Mka_00030]).⌋()

8.3.6 Mka_GetEnable

[CP_SWS_Mka_91017]{DRAFT} ⌈

Service Name	Mka_GetEnable (DRAFT)	
Syntax	`Std_ReturnType Mka_GetEnable ( uint8 MkaPaeIdx, boolean* EnablePtr )`	
Service ID [hex]	0x5	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different MkaPaeIdx, Non reentrant for the same MkaPaeIdx	
Parameters (in)	MkaPaeIdx	Index of the PAE within the context of the MKA module
Parameters (inout)	None	
Parameters (out)	EnablePtr	Pointer to the Mka activation status of a PAE.
Return value	Std_ReturnType	E_OK: The request has been accepted E_NOT_OK: The request has not been accepted
Description	Get the MKA activation status of a PAE **Tags:** atp.Status=DRAFT	
Available via	Mka.h	

⌋()

8.3.7 Mka_GetPaeStatus

[CP_SWS_Mka_91018]{DRAFT} ⌈

Service Name	Mka_GetPaeStatus (DRAFT)
Syntax	`Std_ReturnType Mka_GetPaeStatus ( uint8 MkaPaeIdx, Mka_PaeStatusType* PaeStatusPtr )`

▽

△

Service ID [hex]	0x6	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different MkaPaeIdx, Non reentrant for the same MkaPaeIdx	
Parameters (in)	MkaPaeIdx	Index of the PAE within the context of the MKA module
Parameters (inout)	None	
Parameters (out)	PaeStatusPtr	Pointer to the status structure, which includes detailed information of a PAE.
Return value	Std_ReturnType	E_OK: The request has been accepted E_NOT_OK: The request has not been accepted
Description	Get detailed information of a PAE **Tags:** atp.Status=DRAFT	
Available via	Mka.h	

⌋()

8.3.8 Mka_SetPaePermissiveMode

[CP_SWS_Mka_91021]{DRAFT} ⌈

Service Name	Mka_SetPaePermissiveMode (DRAFT)	
Syntax	`Std_ReturnType Mka_SetPaePermissiveMode ( uint8 MkaPaeIdx, Mka_PermisiveModeType PermisiveMode )`	
Service ID [hex]	0x9	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different MkaPaeIdx, Non reentrant for the same MkaPaeIdx	
Parameters (in)	MkaPaeIdx	Index of the PAE within the context of the MKA module
	PermisiveMode	Permissive mode to set in 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	Set Permissive Mode of a KaY **Tags:** atp.Status=DRAFT	
Available via	Mka.h	

⌋()

[CP_SWS_Mka_01003]{DRAFT} ⌈The function `Mka_SetPaePermissiveMode` shall set the `PermisiveMode` of the `MkaPaeInstance` referred with the `MkaPaeIdx`.
The new `PermisiveMode` shall be persistently stored in NVM and used from next power cycle onwards (as required in [CP_SWS_Mka_00030]).
The per configuration provided `MkaOnFailPermissiveMode` of the `MkaPaeInstance` shall not be used if a valid value is stored on the NVM.⌋()

8.3.9 Mka_StartPae

[CP_SWS_Mka_91022]{DRAFT} ⌈

Service Name	Mka_StartPae (DRAFT)	

Syntax	`Std_ReturnType Mka_StartPae (`
	`  uint8 MkaPaeIdx`
	`)`	

Service ID [hex]	0x10	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different MkaPaeIdx, Non reentrant for the same MkaPaeIdx	
Parameters (in)	MkaPaeIdx	Index of the PAE within the context of the MKA module
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	Manual start of the PAE instace. (In case MkaPaeConfiguration.Autostart = False this method starts the PAE operation) **Tags:** atp.Status=DRAFT	
Available via	Mka.h	

⌋()

[CP_SWS_Mka_01004]{DRAFT} ⌈The function `Mka_StartPae` shall start the operation of the `MkaPaeInstance` referred with the `MkaPaeIdx` if the `MkaAutoStart` configuration parameter is TRUE.
If the `MkaAutoStart` configuration parameter is FALSE, `Mka_StartPae` will not have any effect on the referred `MkaPaeInstance`.⌋()

8.3.10 Mka_GetMacSecStatistics

[CP_SWS_Mka_91019]{DRAFT} ⌈

Service Name	Mka_GetMacSecStatistics (DRAFT)	

Syntax	`Std_ReturnType Mka_GetMacSecStatistics (`
	`  uint8 MkaPaeIdx,`
	`  const uint8* Ckn,`
	`  uint8 CknLength`
	`)`	

Service ID [hex]	0x7	
Sync/Async	Asynchronous	
Reentrancy	Reentrant for different MkaPaeIdx, Non reentrant for the same MkaPaeIdx	
Parameters (in)	MkaPaeIdx	Index of the PAE within the context of the MKA module
	Ckn	Connectivity Association Key Name to identify the KaY participant
	CknLength	Length of the CKN parameter provided
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	Get Statistics of a PAE **Tags:** atp.Status=DRAFT
Available via	Mka.h

⌋()

8.3.11 Mka_LinkStateChange

[CP_SWS_Mka_91023]{DRAFT} ⌈

Service Name	Mka_LinkStateChange (DRAFT)	

Syntax	`Std_ReturnType Mka_LinkStateChange (`
	`  uint8 MkaPaeIdx,`
	`  EthTrcv_LinkStateType TransceiverLinkState`
	`)`	

Service ID [hex]	0x1d	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different MkaPaeIdx, Non reentrant for the same MkaPaeIdx	
Parameters (in)	MkaPaeIdx	Index of the PAE within the context of the MKA module
	TransceiverLinkState	The Ethernet link state of a physical Ethernet connection.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: The request has been accepted E_NOT_OK: The request has not been accepted
Description	To inform MKA that a dedicated Trcv/Switch/PAC port link state has changed **Tags:** atp.Status=DRAFT	
Available via	Mka.h	

⌋()

8.4 Callback notifications

This is a list of functions provided for other modules.

8.4.1 Mka_GetMacSecStatisticsNotification

[CP_SWS_Mka_91024]{DRAFT} ⌈

Service Name	Mka_GetMacSecStatisticsNotification (DRAFT)

Syntax	`void Mka_GetMacSecStatisticsNotification (`
	`  uint8 MkaPaeIdx,`
	`  const Mka_Stats_SecYType* MacSecStatsPtr`
	`)`

▽

△

Service ID [hex]	0x1e	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different MkaPaeIdx, Non reentrant for the same MkaPaeIdx	
Parameters (in)	MkaPaeIdx	Index of the PAE within the context of the MKA module
	MacSecStatsPtr	Pointer to a structure including the MACsec statistics of an MKA participant
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Callback to notify that Mka_GetMacSecStatistics finished and provide the requested statistics. **Tags:** atp.Status=DRAFT	
Available via	Mka.h	

⌋()

8.4.2 Mka_RxIndication

[CP_SWS_Mka_91029]{DRAFT} ⌈

Service Name	Mka_RxIndication (DRAFT)	

Syntax	`void Mka_RxIndication (`
	`  uint8 CtrlIdx,`
	`  Eth_FrameType FrameType,`
	`  boolean IsBroadcast,`
	`  const uint8* PhysAddrPtr,`
	`  const uint8* DataPtr,`
	`  uint16 LenByte`
	`)`	

Service ID [hex]	0x1f	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	CtrlIdx	Index of the physical Ethernet controller within the context of the Ethernet Interface
	FrameType	Frame type of received Ethernet frame
	IsBroadcast	parameter to indicate a broadcast frame
	PhysAddrPtr	Pointer to Physical source address (MAC address in network byte order) of received Ethernet frame
	DataPtr	Pointer to payload of received Ethernet frame.
	LenByte	Length (bytes) of the payload in received frame.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	To inform Mka about the reception of MKA Frames **Tags:** atp.Status=DRAFT	
Available via	Mka.h	

⌋()

8.4.3 Mka_TxConfirmation

[CP_SWS_Mka_91030]{DRAFT} ⌈

Service Name	Mka_TxConfirmation (DRAFT)	
Syntax	`void Mka_TxConfirmation ( uint8 CtrlIdx, Eth_BufIdxType BufIdx, Std_ReturnType Result )`	
Service ID [hex]	0x20	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	CtrlIdx	Index of the physical Ethernet controller within the context of the Ethernet Interface
	BufIdx	Index of the transmitted buffer
	Result	E_OK: The transmission was successful E_NOT_OK: The transmission failed.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	To inform MKA about the correct transmition of MKA Frames. **Tags:** atp.Status=DRAFT	
Available via	Mka.h	

⌋()

8.4.4 Mka_MacSecUpdateSecYNotification

[CP_SWS_Mka_91031]{DRAFT} ⌈

Service Name	Mka_MacSecUpdateSecYNotification (DRAFT)	
Syntax	`void Mka_MacSecUpdateSecYNotification ( uint8 MkaPaeIdx )`	
Service ID [hex]	0x21	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different MkaPaeIdx, Non reentrant for the same MkaPaeIdx	
Parameters (in)	MkaPaeIdx	Index of the PAE within the context of the MKA module
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Callback to notify that EthIf_MacSecUpdateSecY finished. **Tags:** atp.Status=DRAFT	
Available via	Mka.h	

⌋()

8.4.5 Mka_MacSecAddTxSaNotification

[CP_SWS_Mka_91032]{DRAFT} ⌈

Service Name	Mka_MacSecAddTxSaNotification (DRAFT)	
Syntax	`void Mka_MacSecAddTxSaNotification ( uint8 MkaPaeIdx )`	
Service ID [hex]	0x22	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different MkaPaeIdx, Non reentrant for the same MkaPaeIdx	
Parameters (in)	MkaPaeIdx	Index of the PAE within the context of the MKA module
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Callback to notify that EthIf_MacSecAddTxSa finished. **Tags:** atp.Status=DRAFT	
Available via	Mka.h	

⌋()

8.4.6 Mka_MacSecAddRxSaNotification

[CP_SWS_Mka_91033]{DRAFT} ⌈

Service Name	Mka_MacSecAddRxSaNotification (DRAFT)	
Syntax	`void Mka_MacSecAddRxSaNotification ( uint8 MkaPaeIdx )`	
Service ID [hex]	0x23	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different MkaPaeIdx, Non reentrant for the same MkaPaeIdx	
Parameters (in)	MkaPaeIdx	Index of the PAE within the context of the MKA module
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Callback to notify that EthIf_MacSecAddRxSa finished. **Tags:** atp.Status=DRAFT	
Available via	Mka.h	

⌋()

8.5 Scheduled functions

These functions are directly called by Basic Software Scheduler. The following functions shall have no return value and no parameter. All functions shall be non reentrant.

8.5.1 Mka_MainFunction

[CP_SWS_Mka_91034]{DRAFT} ⌈

Service Name	Mka_MainFunction (DRAFT)

Syntax	`void Mka_MainFunction (`
	`  void`
	`)`

Service ID [hex]	0x24
Description	Main function for cyclic call. **Tags:** atp.Status=DRAFT
Available via	Mka.h

⌋()

[CP_SWS_Mka_01005]{DRAFT} ⌈The frequency of invocations of `Mka_MainFunction` is determined by the configuration parameter `MkaMainFunctionPeriod`.⌋()

8.6 Expected interfaces

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

8.6.1 Mandatory interfaces

Note: This section defines all interfaces, which are required to fulfill the core functionality of the module.

[CP_SWS_Mka_91006]{DRAFT} ⌈

API Function	***Header File***	***Description***
EthIf_ProvideTxBuffer	EthIf.h	Provides access to a transmit buffer of the specified Ethernet controller.
EthIf_Transmit	EthIf.h	Triggers transmission of a previously filled transmit buffer
NvM_EraseNvBlock	NvM.h	Service to erase a NV block.
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.

⌋()

8.6.2 Optional interfaces

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

[CP_SWS_Mka_91007]{DRAFT} ⌈

API Function	*Header File*	*Description*
Det_ReportError	Det.h	Service to report development errors.
Det_ReportRuntimeError	Det.h	Service to report runtime errors. If a callout has been configured then this callout shall be called.
EthIf_GetPhysAddr	EthIf.h	Obtains the physical source address used by the indexed controller
EthIf_SetSwitchMgmtInfo	EthIf.h	Provides additional management information along to an Ethernet frame that requires special treatment within the Switch. It has to be called between EthIf_ProvideTxBuffer() and EthIf_Transmit() of the related frame.
IdsM_SetSecurityEvent	IdsM.h	This API is the application interface to report security events to the IdsM.

⌋()

8.6.3 Configurable interfaces

There are no configurable interfaces defined.

8.7 Service Interfaces

There are no service interfaces defined.

9 Sequence diagrams

The sequence diagrams show the basic operations carried out during operation.
The purpose of the sequence diagrams is to depict the expected behavior of the communication stack at a glance.

The communication initialization sequence diagrams illustrate how the MKA module gets involved in the Ethernet stack start-up including upper and lower layer modules.

9.1 Communication initialization with MACsec

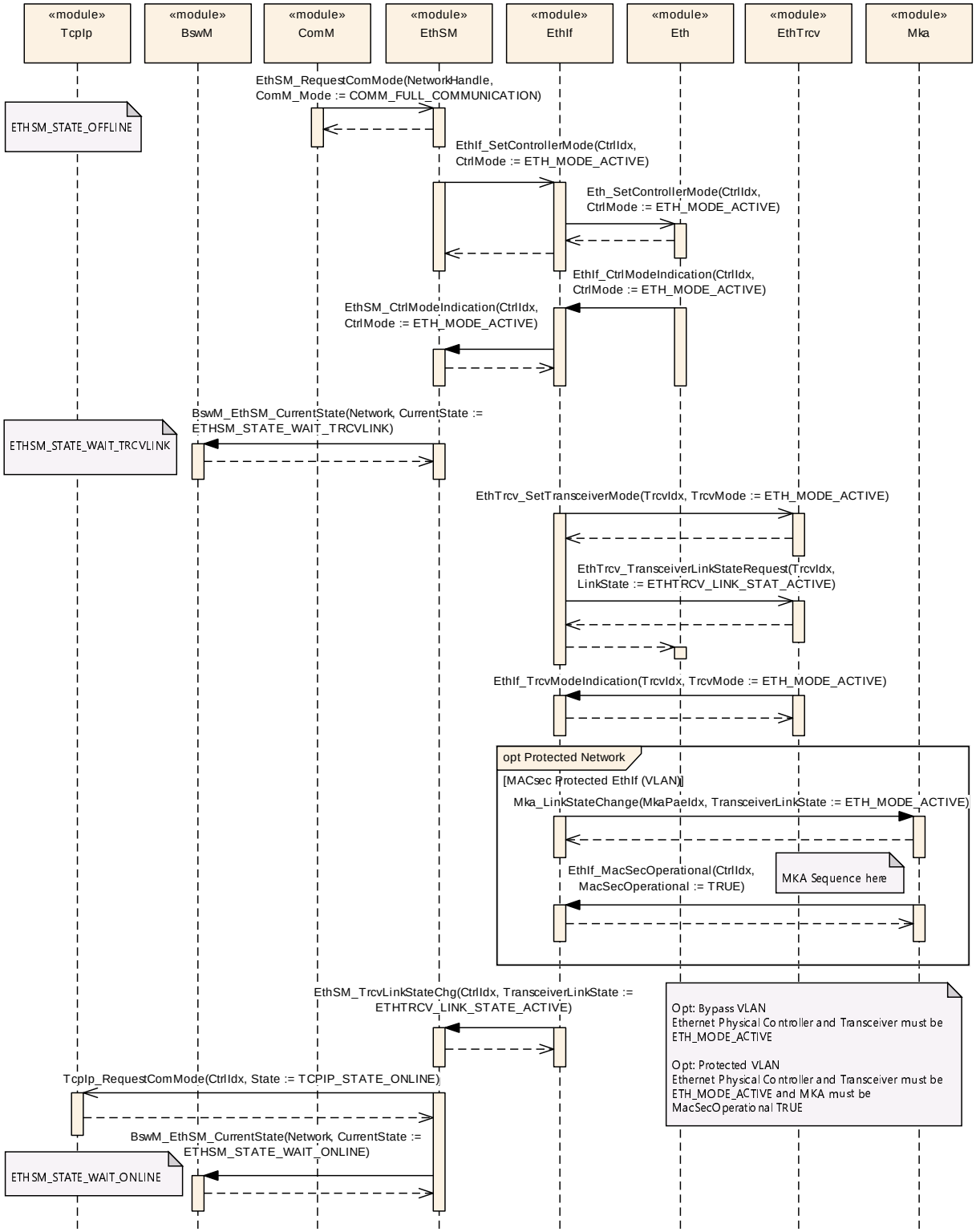

Figure 9.1: Communication initialization with MACsec protected EthIf

9.2 Communication initialization with MACsec and Switch

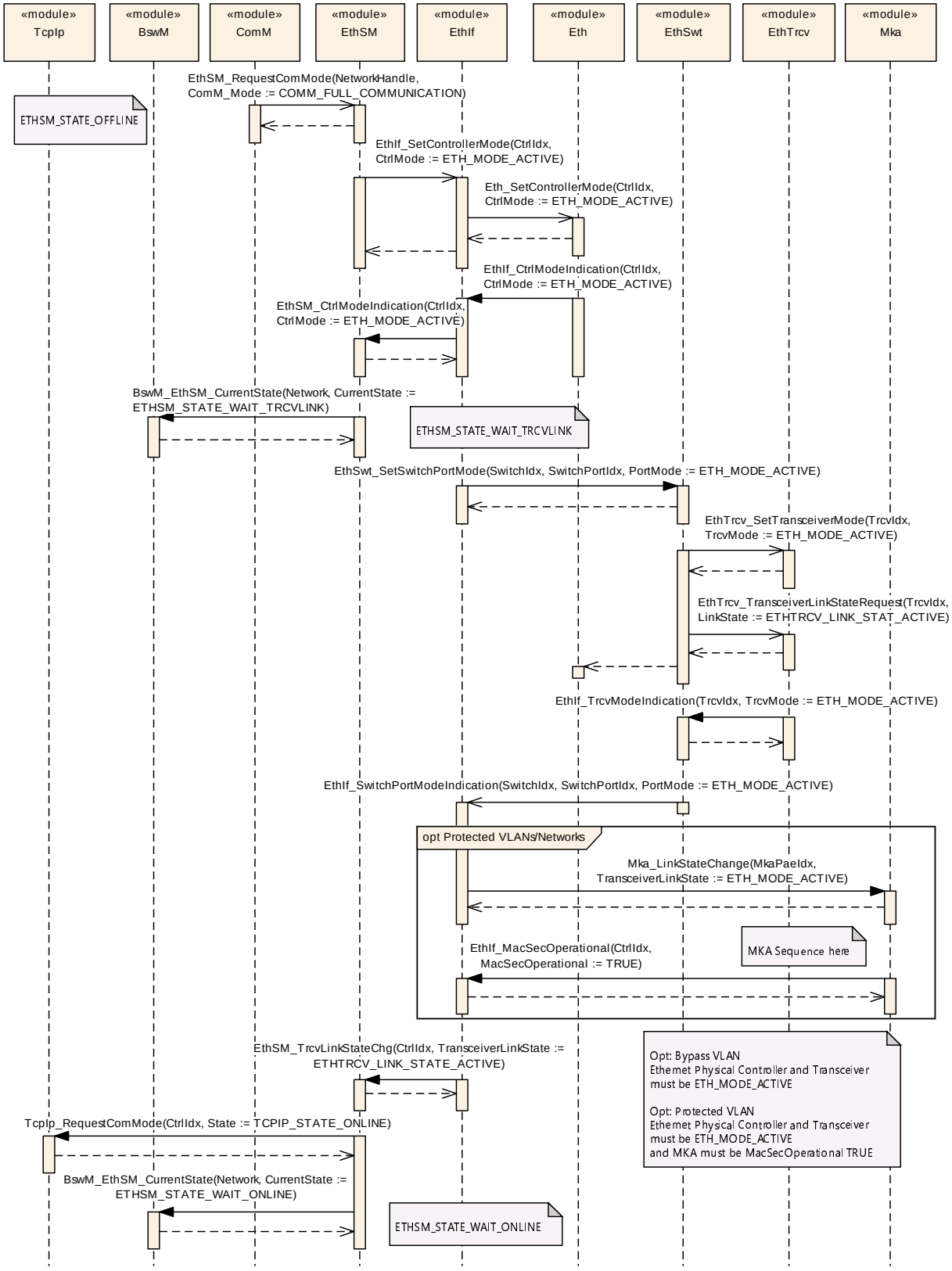

Figure 9.2: Communication initialization with MACsec protected EthIf and Switch

10 Configuration specification

In general, this chapter defines configuration parameters and their clustering into containers. In order to support the specification Chapter 10.1 describes fundamentals. It also specifies a template (table) you shall use for the parameter specification. We intend to leave Chapter 10.1 in the specification to guarantee comprehension.

Chapter 10.2 specifies the structure (containers) and the parameters of the module MKA.

Chapter 10.3 specifies published information of the module MKA.

10.1 How to read this chapter

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

10.2 Containers and configuration parameters

The following chapters summarize all configuration parameters. The detailed meanings of the parameters describe Chapter 7 and Chapter 8.

10.2.1 Mka

SWS Item	[ECUC_Mka_00001]
Module Name	Mka
Description	Configuration of the MACsec Key Agreement module.
Post-Build Variant Support	false
Supported Config Variants	VARIANT-PRE-COMPILE

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
MkaCryptoAlgoConfig	1..255	Cryptography configuration for MACsec. **Tags:** atp.Status=draft
MkaGeneral	1	This container holds the general parameters of the MKA protocol which apply to ports that are referencing this container. **Tags:** atp.Status=draft
MkaPaeConfiguration	1..255	Common MKA configuration for a PAE. **Tags:** atp.Status=draft
MkaPaeInstance	1..255	MKA configuration of a controlled port. **Tags:** atp.Status=draft

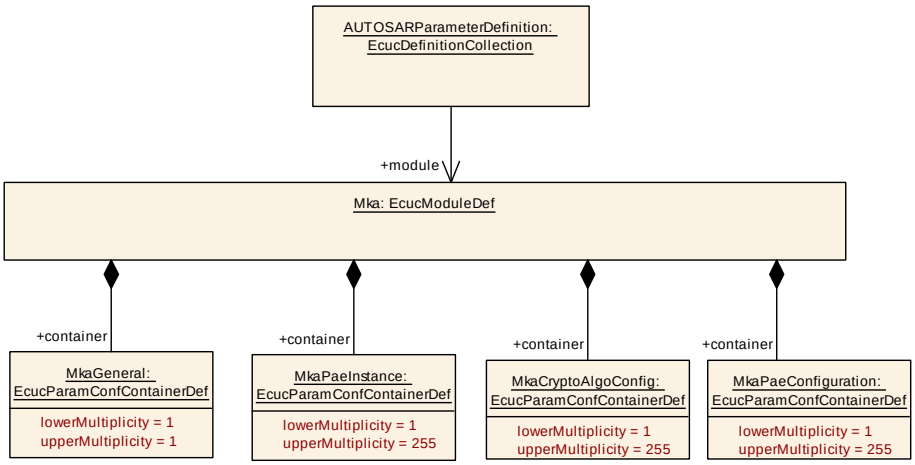

Figure 10.1: Mka

10.2.2 MkaGeneral

SWS Item	**[ECUC_Mka_00002]**
Container Name	MkaGeneral
Parent Container	Mka
Description	This container holds the general parameters of the MKA protocol which apply to ports that are referencing this container. **Tags:** atp.Status=draft
Configuration Parameters	

SWS Item	**[ECUC_Mka_00034]**		
Parameter Name	MkaDevErrorDetect		
Parent Container	MkaGeneral		
Description	Switches the development error detection and notification on or off. - true: detection and notification is enabled. . false: detection and notification is disabled. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_Mka_00007]**
Parameter Name	MkaHelloTime
Parent Container	MkaGeneral
Description	Interval [s] between MKPDUs when two participants have an active MKA communication (the participants are included in the Live Peer list of each other). **Tags:** atp.Status=draft
Multiplicity	1

△

Type	EcucFloatParamDef		
Range	[0 .. INF]		
Default value	2		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_Mka_00009]**		
Parameter Name	MkaLifeTime		
Parent Container	MkaGeneral		
Description	Time span [s] since last MKPDU was received from the other participant, to consider it alive. \newline In case no valid MKPDU from the other participant is received after Mka LifeTime, the Secure Channel is shut down. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	[0 .. INF]		
Default value	6		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_Mka_00035]**		
Parameter Name	MkaMainFunctionPeriod		
Parent Container	MkaGeneral		
Description	The cycle time of the periodic main function of MKA. Defined in seconds. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	[0 .. INF]		
Default value	–		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_Mka_00010]**		
Parameter Name	MkaSakRetireTime		
Parent Container	MkaGeneral		
Description	During an SAK rekey, time [s] to retire the previous SAK in use. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	[0 .. INF]		
Default value	3		

▽

△

Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_Mka_00036]**		
Parameter Name	MkaVersionInfoApi		
Parent Container	MkaGeneral		
Description	Enables / Disables version info API. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

No Included Containers

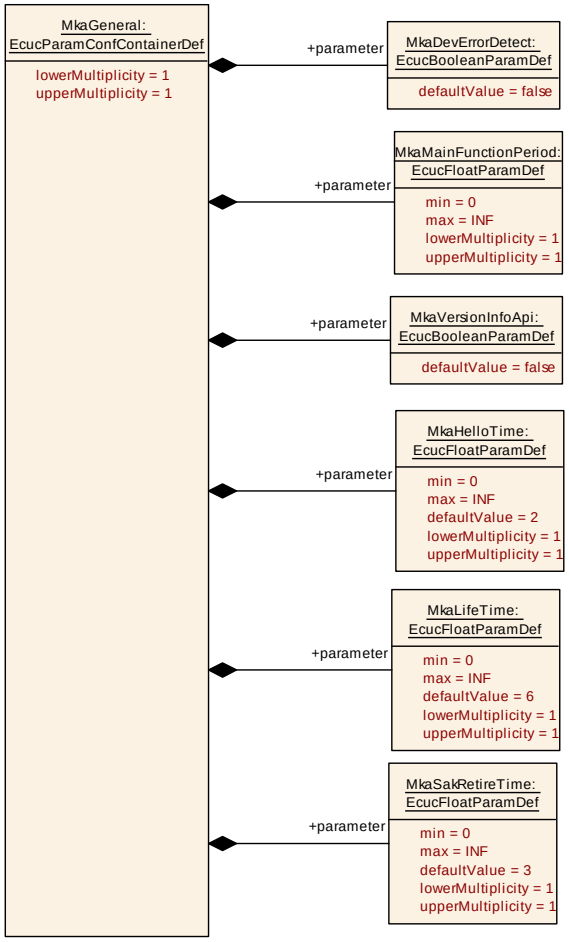

Figure 10.2: MkaGeneral

10.2.3 MkaPaeConfiguration

SWS Item	[ECUC_Mka_00033]		
Container Name	MkaPaeConfiguration		
Parent Container	Mka		
Description	Common MKA configuration for a PAE. **Tags:** atp.Status=draft		
Configuration Parameters			

SWS Item	[ECUC_Mka_00012]		
Parameter Name	MkaAutoStart		
Parent Container	MkaPaeConfiguration		
Description	Autostart or manual start of the PAE Instance. True := Autostart False := Manual Start If Autostart = False, the method Mka_StartPae is used to start the PAE instance. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	true		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

SWS Item	[ECUC_Mka_00037]		
Parameter Name	MkaPaeConfigurationIdx		
Parent Container	MkaPaeConfiguration		
Description	Instance ID of the MkaPaeConfiguration. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 255		
Default value	–		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

SWS Item	[ECUC_Mka_00004]	
Parameter Name	MkaRetryBaseDelay	
Parent Container	MkaPaeConfiguration	
Description	The base delay in seconds for the retry phase of MKA. The retry have an exponential back off delay (1x base delay, 2x base delay, 4x base delay, ...) until the retry delay overflows the MkaRetryCyclicDelay value. **Tags:** atp.Status=draft	
Multiplicity	1	
Type	EcucFloatParamDef	
Range	[0 .. INF]	

△

Default value	–		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_Mka_00005]**		
Parameter Name	MkaRetryCyclicDelay		
Parent Container	MkaPaeConfiguration		
Description	Interval in seconds between retries after base delay with exponential back off. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	[0 .. INF]		
Default value	–		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_Mka_00024]**		
Parameter Name	MkaSakRekeyTimeSpan		
Parent Container	MkaPaeConfiguration		
Description	Time [s] to trigger the rekey of an in use SAK. If set to 0, the rekey will not be triggered after a time span. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	[0 .. INF]		
Default value	–		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

No Included Containers

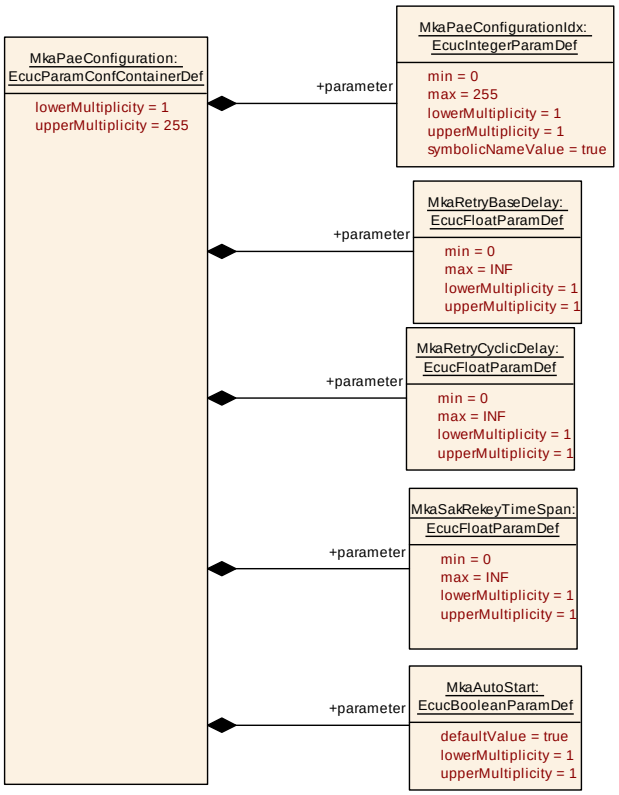

Figure 10.3: MkaPaeConfiguration

10.2.4 MkaCryptoAlgoConfig

SWS Item	**[ECUC_Mka_00021]**
Container Name	MkaCryptoAlgoConfig
Parent Container	Mka
Description	Cryptography configuration for MACsec. **Tags:** atp.Status=draft
Configuration Parameters	

SWS Item	**[ECUC_Mka_00053]**		
Parameter Name	MkaCryptoAlgoConfigIdx		
Parent Container	MkaCryptoAlgoConfig		
Description	Instance ID of the configured Crypto configuration. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 255		
Default value	–		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	

▽

△

	Post-build time	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_Mka_00025]**		
Parameter Name	MkaMacSecCapability		
Parent Container	MkaCryptoAlgoConfig		
Description	MACsec capability to use for MACsec. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	INTEGRITY_AND_ CONFIDENTIALITY	– **Tags:** atp.Status=draft	
	INTEGRITY_WITHOUT_ CONFIDENTIALITY	– **Tags:** atp.Status=draft	
Default value	INTEGRITY_WITHOUT_CONFIDENTIALITY		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_Mka_00026]**		
Parameter Name	MkaMacSecConfidentialityOffset		
Parent Container	MkaCryptoAlgoConfig		
Description	The confidentiality Offset is only applicable if "Integrity and confidentiality" with a non-XPN cipher suite is selected. **Tags:** atp.Status=draft		
Multiplicity	0..1		
Type	EcucEnumerationParamDef		
Range	CONFIDENTIALITY_OFFSET_0	– **Tags:** atp.Status=draft	
	CONFIDENTIALITY_OFFSET_30	– **Tags:** atp.Status=draft	
	CONFIDENTIALITY_OFFSET_50	– **Tags:** atp.Status=draft	
Default value	CONFIDENTIALITY_OFFSET_0		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_Mka_00027]**
Parameter Name	MkaMacSecReplayProtection
Parent Container	MkaCryptoAlgoConfig

▽

△

Description	MACsec replay protection parameter for MACsec.\newline The Replay Protection parameter is defined in the IEEE 802.1AE-2018 document, on chapter 10.4. It enables the replay protection if a packet is received with PacketNumber outside of the Window = PN - ProtectionWindow. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_Mka_00028]**		
Parameter Name	MkaMacSecReplayProtectionWindow		
Parent Container	MkaCryptoAlgoConfig		
Description	In case replay protection is active, replay protection window.The Protection Window is a positive integer between 0 and 2^32-1 (No XPN) or 2^30-1 (XPN). **Tags:** atp.Status=draft		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 18446744073709551615		
Default value	–		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
MkaCipherSuites	1..4	Cipher Suite configuration to use with MACsec. MkaCipherSuite Prio is present in case the MKA instance acts as a Key Server to select the cipher suite to use for MACsec.

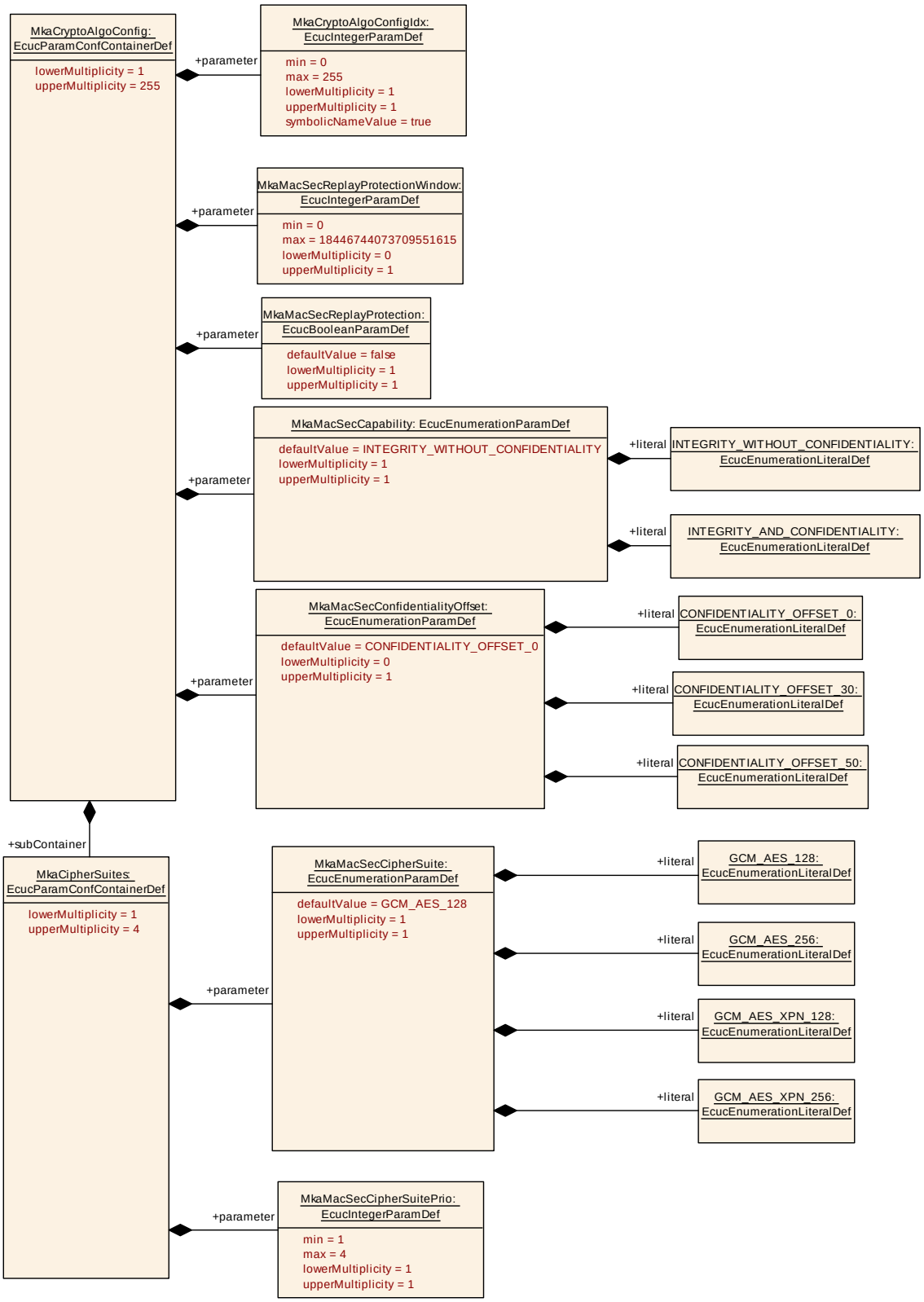

Figure 10.4: MkaCryptoAlgoConfig

10.2.5 MkaCipherSuites

SWS Item	[ECUC_Mka_00050]		
Container Name	MkaCipherSuites		
Parent Container	MkaCryptoAlgoConfig		
Description	Cipher Suite configuration to use with MACsec. MkaCipherSuitePrio is present in case the MKA instance acts as a Key Server to select the cipher suite to use for MACsec.		
Configuration Parameters			

SWS Item	[ECUC_Mka_00052]		
Parameter Name	MkaMacSecCipherSuite		
Parent Container	MkaCipherSuites		
Description	Cipher Suite to use for MACsec. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	GCM_AES_128	– **Tags:** atp.Status=draft	
	GCM_AES_256	– **Tags:** atp.Status=draft	
	GCM_AES_XPN_128	– **Tags:** atp.Status=draft	
	GCM_AES_XPN_256	– **Tags:** atp.Status=draft	
Default value	GCM_AES_128		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

SWS Item	[ECUC_Mka_00051]		
Parameter Name	MkaMacSecCipherSuitePrio		
Parent Container	MkaCipherSuites		
Description	In case the MKA instance acts as a Key Server, the priority is used to select the Cipher Suite to use with MACsec from the common supported Ciphers (with the client in the link). Value of 1 means the highest priority. Value of 4 means the lowest priority. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 4		
Default value	–		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

No Included Containers

10.2.6 MkaPaeInstance

SWS Item	[ECUC_Mka_00003]
Container Name	MkaPaeInstance
Parent Container	Mka
Description	MKA configuration of a controlled port. **Tags:** atp.Status=draft
Configuration Parameters	

SWS Item	[ECUC_Mka_00018]		
Parameter Name	MkaOnFailPermissiveMode		
Parent Container	MkaPaeInstance		
Description	Sets the behavior of the PAE in case MKA does not succeed when MKA is enabled. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	NEVER	The controlled port will never be set to enabled if the participants cannot establish and successfully use a MACsec Secure Channel. **Tags:** atp.Status=draft	
	TIMEOUT	The controlled port will be set to enabled and MACsec will not be used in the referred port if the timeout value (MkaOnFailPermissiveMode Timeout) is reached and none MKA instance under the PAE instance could success the following conditions: - A participant belonging to the same CA was recognized and authenticated. - A secure channel could be established. - Both participants can transmit and receive MACsec protected traffic through the SC. **Tags:** atp.Status=draft	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

SWS Item	[ECUC_Mka_00019]		
Parameter Name	MkaOnFailPermissiveModeTimeout		
Parent Container	MkaPaeInstance		
Description	Timeout in seconds to enable the controlled port in case MkaOnFailPermissiveMode is set to Timeout. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	[0 .. INF]		
Default value	255		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

SWS Item	[ECUC_Mka_00011]		
Parameter Name	MkaPaeIdx		
Parent Container	MkaPaeInstance		
Description	Instance ID of the configured PAE. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 255		
Default value	–		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

SWS Item	[ECUC_Mka_00013]		
Parameter Name	MkaEthIfControllerRef		
Parent Container	MkaPaeInstance		
Description	A reference to the EthIfController which is used for transmitting / receiving EAP frames (to configure the controlled port). **Tags:** atp.Status=draft		
Multiplicity	1		
Type	Symbolic name reference to EthIfController		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

SWS Item	[ECUC_Mka_00054]		
Parameter Name	MkaPaeConfRef		
Parent Container	MkaPaeInstance		
Description	Reference to the applicable PAE configuration. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	Reference to MkaPaeConfiguration		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

SWS Item	[ECUC_Mka_00014]
Parameter Name	MkaSwitchPortRef
Parent Container	MkaPaeInstance
Description	A reference to the EthSwtPort enabled and set only in case PAE is attached to a switch port. **Tags:** atp.Status=draft
Multiplicity	0..1
Type	Symbolic name reference to EthSwtPort

▽

△

Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
MkaKay	1	MKA instance (KaY) for a controlled port (PaE). **Tags:** atp.Status=draft

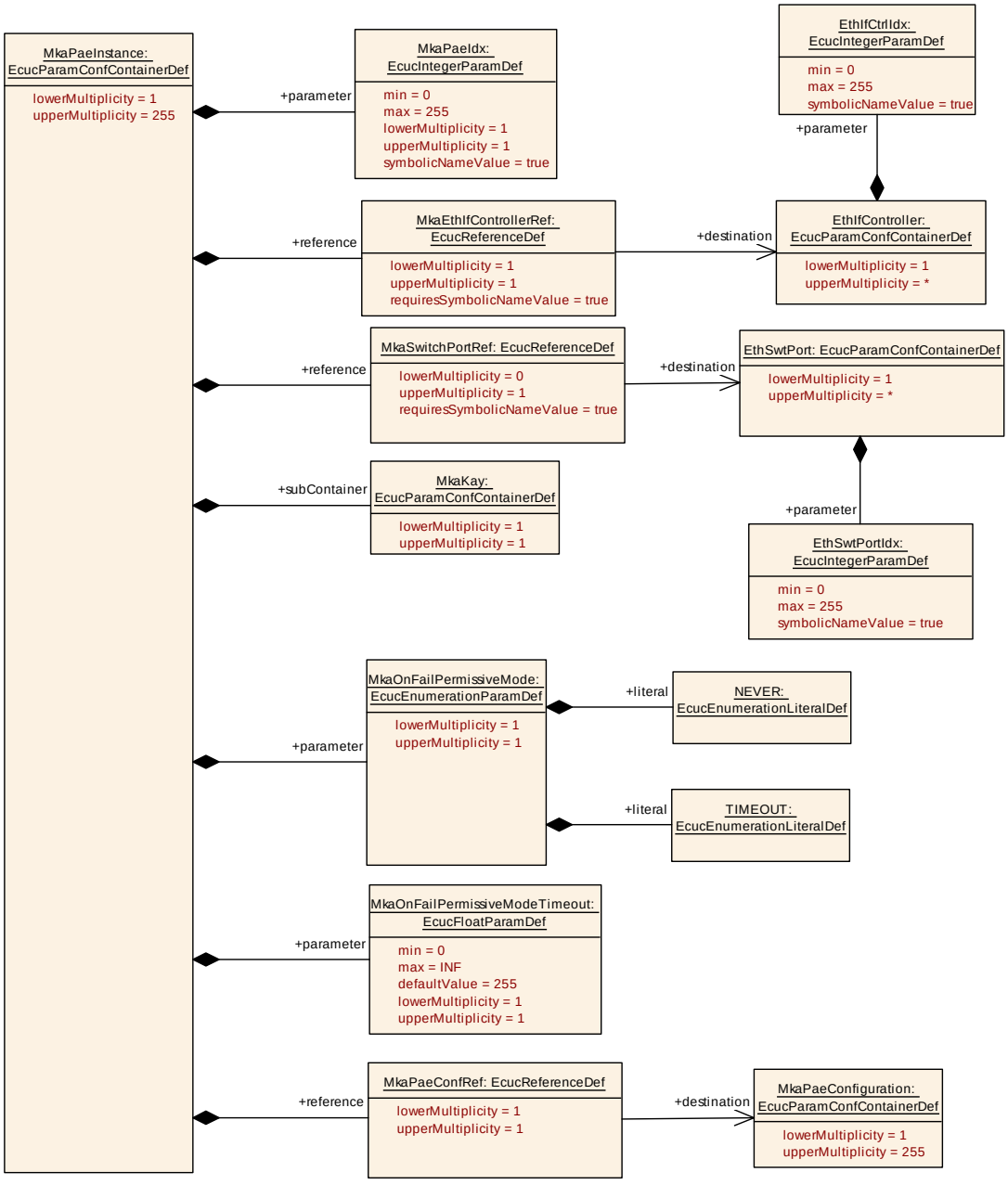

Figure 10.5: MkaPaeInstance

10.2.7 MkaKay

SWS Item	[ECUC_Mka_00017]		
Container Name	MkaKay		
Parent Container	MkaPaeInstance		
Description	MKA instance (KaY) for a controlled port (PaE). **Tags:** atp.Status=draft		
Configuration Parameters			

SWS Item	[ECUC_Mka_00016]		
Parameter Name	MkaBypassEtherType		
Parent Container	MkaKay		
Description	Bypassed EtherType. The EtherTypes included will not be MACsec protected. **Tags:** atp.Status=draft		
Multiplicity	0..255		
Type	EcucIntegerParamDef		
Range	0 .. 65535		
Default value	–		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

SWS Item	[ECUC_Mka_00015]		
Parameter Name	MkaBypassVlan		
Parent Container	MkaKay		
Description	Bypassed VLAN-ID. The VLAN-IDs included will not be MACsec protected. (VLAN-ID 0 is interpreted as no-VLAN –> Bypass untagged traffic) **Tags:** atp.Status=draft		
Multiplicity	0..255		
Type	EcucIntegerParamDef		
Range	0 .. 4094		
Default value	–		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

SWS Item	[ECUC_Mka_00032]
Parameter Name	MkaDstMacAddress
Parent Container	MkaKay
Description	Destination MAC address to use by the MKA instance. The destination MAC addresses to use are defined in the IEEE 802.1X-2020 chapter 11.1.1 (Table 11-1). **Tags:** atp.Status=draft
Multiplicity	1
Type	EcucStringParamDef
Default value	–

△

Regular Expression	–		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_Mka_00022]**		
Parameter Name	MkaKeyServerPriority		
Parent Container	MkaKay		
Description	Key Server Priority of the MKA participants. In case it is not provided, the default value is 0 for an MKA_KEY_SERVER and 255 for an MKA_PEER. **Tags:** atp.Status=draft		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default value	–		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_Mka_00029]**		
Parameter Name	MkaRole		
Parent Container	MkaKay		
Description	Role of the MKA instance. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	MKA_KEY_SERVER	– **Tags:** atp.Status=draft	
	MKA_PEER	– **Tags:** atp.Status=draft	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_Mka_00031]**		
Parameter Name	MkaSrcMacAddress		
Parent Container	MkaKay		
Description	Source MAC address to use by the MKA instance. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucStringParamDef		
Default value	–		
Regular Expression	–		

▽

△

Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
MkaKayDemEventParameterRefs	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. **Tags:** atp.Status=draft
MkaKayParticipant	1..255	MKA participant configuration. **Tags:** atp.Status=draft

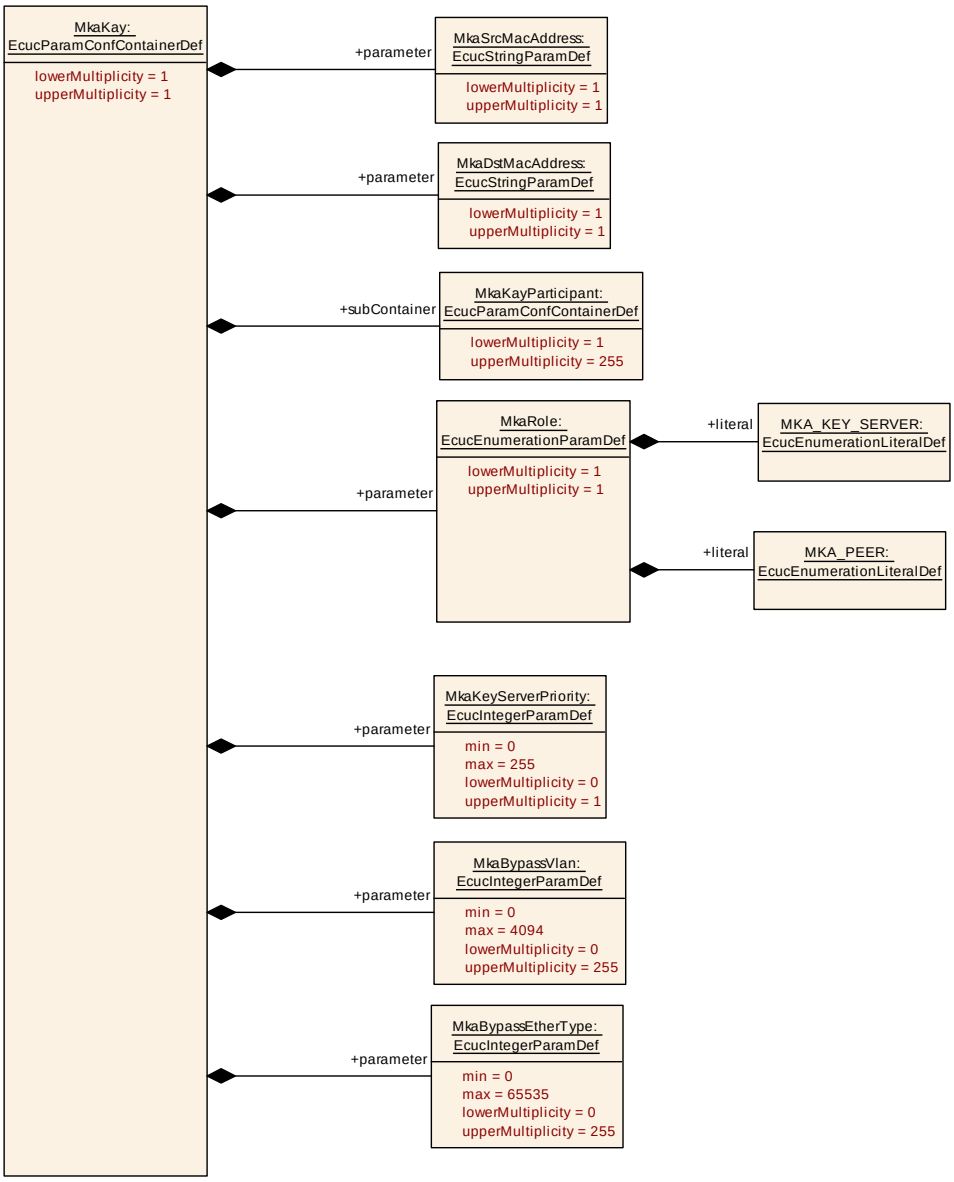

Figure 10.6: MkaKay

10.2.8 MkaKayParticipant

SWS Item	**[ECUC_Mka_00038]**
Container Name	MkaKayParticipant
Parent Container	MkaKay
Description	MKA participant configuration. **Tags:** atp.Status=draft
Configuration Parameters	

SWS Item	[ECUC_Mka_00049]		
Parameter Name	MkaParticipantActivate		
Parent Container	MkaKayParticipant		
Description	Enabled/Disabled status of the MKA participant. - True = The MKA Participant exchanges MKPDUs - False = The MKA participant does not exchange MKPDUs. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

SWS Item	[ECUC_Mka_00048]		
Parameter Name	MkaCryptoAlgoRef		
Parent Container	MkaKayParticipant		
Description	Reference to the cryptography to use (MkaAlgoConfiguration Container). **Tags:** atp.Status=draft		
Multiplicity	1		
Type	Reference to MkaCryptoAlgoConfig		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

SWS Item	[ECUC_Mka_00040]		
Parameter Name	MkaCryptoCknCakKeyRef		
Parent Container	MkaKayParticipant		
Description	Reference to the CKN (min. 1 & max. 32 characters) assigned to the KaY Participant in the CSM. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	Symbolic name reference to CsmKey		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

SWS Item	[ECUC_Mka_00042]
Parameter Name	MkaCryptoIckDeriveJobRef
Parent Container	MkaKayParticipant
Description	Reference to a CSM job for ICK Derivation. **Tags:** atp.Status=draft
Multiplicity	1
Type	Symbolic name reference to CsmJob

▽

△

Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_Mka_00043]**		
Parameter Name	MkaCryptoIcvGenerateJobRef		
Parent Container	MkaKayParticipant		
Description	Reference to a CSM job for ICV generation (according to IEEE_802.x ICV is always 128 bits). **Tags:** atp.Status=draft		
Multiplicity	1		
Type	Symbolic name reference to CsmJob		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_Mka_00044]**		
Parameter Name	MkaCryptoIcvVerifyJobRef		
Parent Container	MkaKayParticipant		
Description	Reference to a CSM job for ICV verification (according to IEEE_802.x ICV is always 128 bits). **Tags:** atp.Status=draft		
Multiplicity	1		
Type	Symbolic name reference to CsmJob		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_Mka_00045]**		
Parameter Name	MkaCryptoKekDeriveJobRef		
Parent Container	MkaKayParticipant		
Description	Reference to a CSM job for KEK Derivation. (Note: CAK needs to be set as the KEK Derive job CsmJobKeyRef) **Tags:** atp.Status=draft		
Multiplicity	1		
Type	Symbolic name reference to CsmJob		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

SWS Item	[ECUC_Mka_00060]		
Parameter Name	MkaCryptoKeyUnwrapJobRef		
Parent Container	MkaKayParticipant		
Description	Reference to a CSM job for SAK unwrap (to perform the Decrypt part of RFC3394). **Tags:** atp.Status=draft		
Multiplicity	1		
Type	Symbolic name reference to CsmJob		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

SWS Item	[ECUC_Mka_00047]		
Parameter Name	MkaCryptoKeyWrapJobRef		
Parent Container	MkaKayParticipant		
Description	Reference to a CSM job for SAK wrap (to perform the Encrypt part of RFC3394). **Tags:** atp.Status=draft		
Multiplicity	1		
Type	Symbolic name reference to CsmJob		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

SWS Item	[ECUC_Mka_00041]		
Parameter Name	MkaCryptoRandomJobRef		
Parent Container	MkaKayParticipant		
Description	Reference to a CSM job for random number generation. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	Symbolic name reference to CsmJob		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

SWS Item	[ECUC_Mka_00046]		
Parameter Name	MkaCryptoSakKeyRef		
Parent Container	MkaKayParticipant		
Description	Reference to a CSM key where SAK shall be stored. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	Symbolic name reference to CsmKey		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	

△

Scope / Dependency	scope: local

No Included Containers

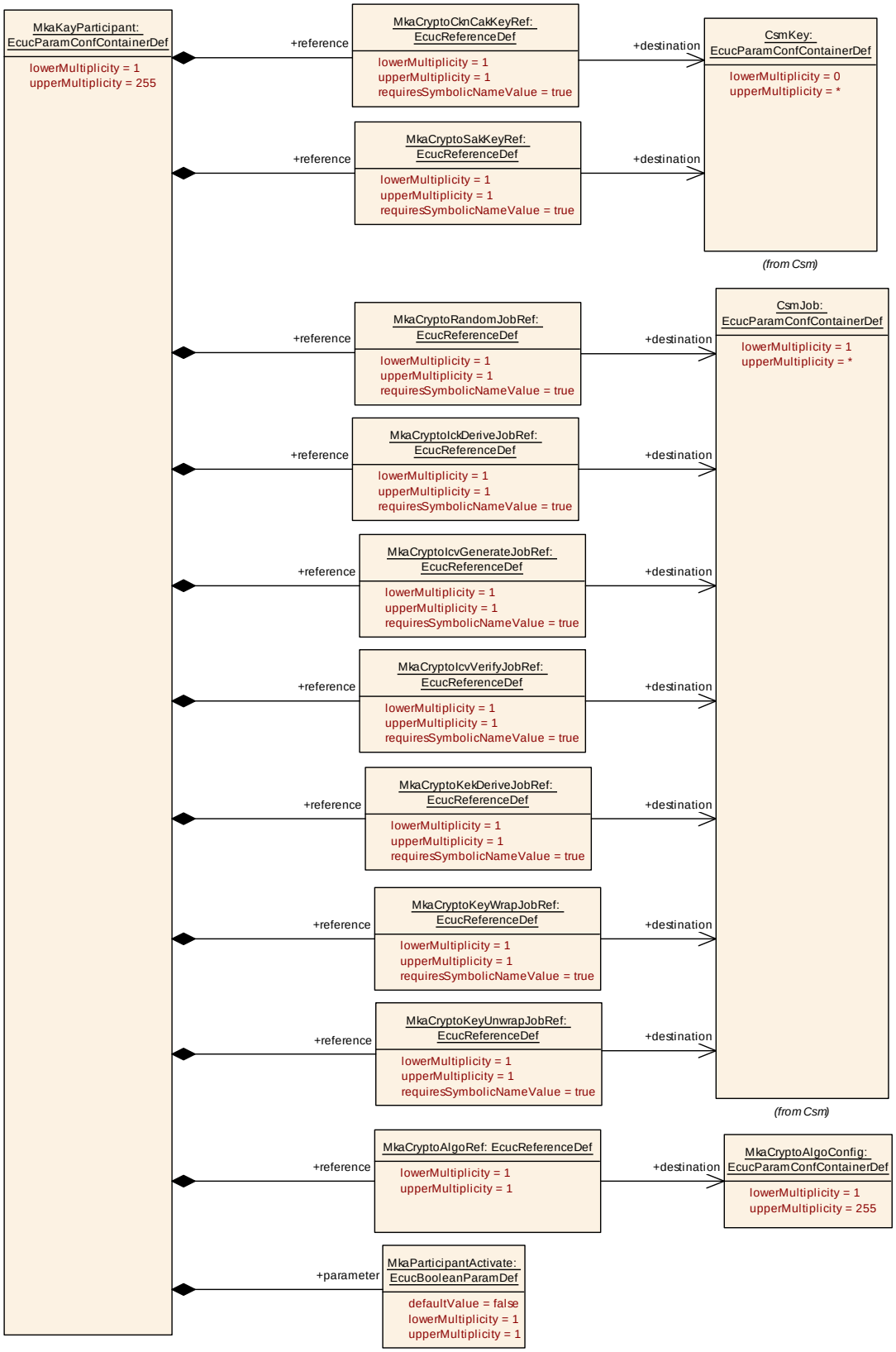

Figure 10.7: MkaKayParticipant

10.2.9 MkaKayDemEventParameterRefs

SWS Item	[ECUC_Mka_00055]
Container Name	MkaKayDemEventParameterRefs
Parent Container	MkaKay
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. **Tags:** atp.Status=draft
Configuration Parameters	

SWS Item	[ECUC_Mka_00059]		
Parameter Name	MKA_E_ALGO_MISMATCH_INSTANCE		
Parent Container	MkaKayDemEventParameterRefs		
Description	Reference to the DemEventParameter which shall be issued when the MkaKay Instance does not successfully agree on MACsec keys and at least one MkaKay Participant does not support a common MACsec cipher suite. **Tags:** atp.Status=draft		
Multiplicity	0..1		
Type	Symbolic name reference to DemEventParameter		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

SWS Item	[ECUC_Mka_00058]		
Parameter Name	MKA_E_KEY_MISMATCH_INSTANCE		
Parent Container	MkaKayDemEventParameterRefs		
Description	Reference to the DemEventParameter which shall be issued when the MkaKay Instance does not successfully agree on MACsec keys and at least one exchange for this MkaKay Instance encounters an ICV validation failure. **Tags:** atp.Status=draft		
Multiplicity	0..1		
Type	Symbolic name reference to DemEventParameter		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	

△

Scope / Dependency	scope: local

SWS Item	**[ECUC_Mka_00057]**		
Parameter Name	MKA_E_KEY_NOT_PRESENT_INSTANCE		
Parent Container	MkaKayDemEventParameterRefs		
Description	Reference to the DemEventParameter which shall be issued when the MkaKay Instance does not successfully agree on MACsec keys and at least one of the keys (CAK) for this MkaKay Instance is not present. **Tags:** atp.Status=draft		
Multiplicity	0..1		
Type	Symbolic name reference to DemEventParameter		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_Mka_00056]**		
Parameter Name	MKA_E_TIMEOUT_INSTANCE		
Parent Container	MkaKayDemEventParameterRefs		
Description	Reference to the DemEventParameter which shall be issued when the MkaKay Instance does not successfully agree on MACsec keys and at least one exchange for this MkaKay Instance encounters a timeout. **Tags:** atp.Status=draft		
Multiplicity	0..1		
Type	Symbolic name reference to DemEventParameter		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

No Included Containers

10.3 Published Information

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

A Not applicable requirements

[CP_SWS_Mka_00999] ⌈These requirements are not applicable to this specification.⌋ *(FO_RS_MACsec_00001, FO_RS_MACsec_00006, FO_RS_MACsec_00011, FO_RS_MACsec_00012, FO_RS_MACsec_00018, FO_RS_MACsec_00019, FO_RS_MACsec_00021, FO_RS_MACsec_00022, FO_RS_MACsec_00034, FO_RS_MACsec_00036)*