VOLKSWAGEN

AKTIENGESELLSCHAFT

Management of SecOC Freshness Values for safer on-board communication

Technical development, specifications:

First edition	19.12.2016
Change status	26.04.2022
Specification version	2.2

Table of contents

1 General 4
1.1 Purpose 4
1.2 Abbreviations 4
1.3 Definitions 4
2 The SOK Freshness Manager 5
2.1 General requirements 5
2.2 Statemachine 5
2.2.1 Constants 5
2.2.1.1 SOK_freshness_value_config_array 8
2.2.1.2 [SOK time server] SOK_time_client_config_array 10
2.2.2 Runtime status 10
2.2.2.1 [SOK participant] SOK_freshness_value_status_array 11
2.3 Signal inputs 13
2.3.1 SokFm_CrReceiveChallenge<SOK_freshness_name> 13
2.4 Signal outputs 14
2.5 Service functions 14
2.5.1 SokFm_MainFunction() 14
2.5.2 SokFm_Deinit() 15
2.5.3 SokFm_TriggerCrRequest*() 16
2.5.4 SokFm_SecOC_VerificationStatusCallout() 16
2.5.5 SokFm_Init() 18
2.6 Callback functions 18
2.6.1 SecOC_GetTxFreshness() 18
2.6.2 SecOC_SPduTxConfirmation() 19
2.6.3 SecOC_GetRxFreshnessAuthData() 19
2.7 Internal interfaces 20
2.7.1 [SOK time server] VW_SOK_FM_initTime() 20
2.7.2 VW_SOK_FM_incClockCount() 21
2.7.3 [SOK participant] VW_SOK_FM_incTimeTimeout() 21
2.7.4 [SOK participant] VW_SOK_FM_receiveUnauthenticatedTime() 22
2.7.5 [SOK participant] VW_SOK_FM_requestAuthenticatedTime() 22
2.7.6 [SOK participant] VW_SOK_FM_receiveAuthenticatedTime() 22
2.7.7 [SOK participant] VW_SOK_FM_getTxTime() 23
2.7.8 [SOK participant] VW_SOK_FM_getRxTime() 24
2.7.9 VW_SOK_FM_getTxChallenge() 26
2.7.10 VW_SOK_FM_getRxChallenge() 27
2.7.11 [SOK time server] VW_SOK_FM_sendUnauthenticatedTime() 28
2.7.12 [SOK time server] VW_SOK_FM_queueAuthenticTimeResponse*() 28
2.7.13 [SOK time server] VW_SOK_FM_sendAuthenticTimeResponse() 29
2.7.14 VW_SOK_FM_deactivate_SOK 29
2.7.15 VW_SOK_FM_activate_SOK 31
2.8 Integration against VKMS 32
2.9 Measured values 33
2.9.1 SOK_verification_failed_list 33
2.9.2 SOK_signature_failed_list 33
2.9.3 SOK_general_information 33
2.9.4 SOK_time_information 34
2.9.5 SOK_freshness_information 34
2.9.6 SOK_missing_key_list 34
2.10 Diagnostic routines 34

2.10.1 SOK_function_deactivation 34
2.10.1.1 Routine_start 34
2.10.1.2 Request_routine_results 35
2.11 Event log entries 35
2.11.1 SOK_timeserver_not_available 36
2.11.2 SOK_signature_verification_failed 36
2.11.3 SOK_key_not_available 37
2.11.4 SOK_function_deactivated 37
3 Appendix 38
3.1 Change documentation 38
3.2 Referenced documents 40

1 General

1.1 Purpose

[I: SOK_FM_7]

This document describes the Freshness Manager requirements of secure on-board communication.

[A: SOK_FM_431]

An implementation of SOK based on this document results in an implementation that conforms to the protocol description of the <protocol description> document.

[A: SOK_FM_639]

The SecOC module from AUTOSAR 4.3 serves as the basis for the implementation.

1.2 Abbreviations

[I: SOK_FM_21]

Table: Abbreviations

Abbreviation	Description
QLAH	Question load book
SOK	Secure onboard communication
SOK-participants	A control device that sends models protected by SOK or empiles
VKMS	Vehicle Key Management System

1.3 Definitions

[I: SOK_FM_23]

Table: Definitions

Bezeichnung	Definition
Date message	The new message which contains the date to be protected; "AuthenticI-PDU" in AUTOSAR Context
Protection message	The new message which contains the signature about the new messages to be protected. contains; "Cryptographic I-PDU" in AUTOSAR-Kontext.
Verwortable function	The function, which allows the transmission of date in the vehicle and determines which of its data mustbe protected with SOK.

2 The SOK Freshness Manager

2.1 General requirements

[A: SOK_FM_739]

The SOK Freshness Manager ("SOK-FM") is responsible for managing the freshness values required by SecOC.

[A: SOK_FM_884]

The SOK-FM is an AUTOSAR 4.3 Complex Device Driver.

[A: SOK_FM_885]

The SOK-FM is developed according to MISRA-C.

[A: SOK_FM_1205]

The SOK-FM must meet the requirements for ASIL-D.

[A: SOK_FM_479]

The SOK-FM is responsible for the execution of the SOK protocols challenge-response and authentic time distribution.

[A: SOK_FM_849]

In the following, the behavior of the SOK-FM is described on the basis of a state machine and functions acting on it. These state machines and functions only serve as an example to illustrate the intended behavior. The actual implementation can deviate from this as long as the behavior of the implementation corresponds to the behavior described here.
However, the existing configuration options must remain as long as they cannot be derived from the configuration of other AUTOSAR components (e.g. CSM and SecOC). Their naming may differ.

[A: SOK_FM_1196]

As return value of all functions the StdReturnType with the following specific error values is to be used:

Value (Hex)	Name
0x00	E_OK
0x01	E_NOT_OK
0x20	SOKFM_ERROR_RNG
0x21	SOKFM_ERROR_SEND
0x30	VW_SOK_ERROR_DEACTIVATING_VERIFICATION_FAILED
0x31	VW_SOK_ERROR_ACTIVATING_VERIFICATION_FAILED
0x32	VW_SOK_ERROR_ACTIVATING_DEFAULT_AUTHENTICATION_FAILED
0x33	VW_SOK_ERROR_DEACTIVATING_DEFAULT_AUTHENTICATION_FAILED
0x34	VW_SOK_ERROR_MUTIPLE_FAILURES
0x35	VW_SOK_ ERROR_PERSISTING_DEACTIVATION_STATUS_FAILED

2.2 Statemachine

2.2.1 Constants

[A: SOK_FM_770]

This section describes the constants that influence the behavior of the SOK-FM.

[A: SOK_FM_771]

These constants must be configurable within the described framework.

[A: SOK_FM_766]

SOK_main_period
- Data type: uint8_t
- *Value range:* Including 5 up to and including 250 in increments of 5
- Default value: 5
- *Description:* The call frequency of the SOK-FM main function in milliseconds.

[A: SOK_FM_901]

SOK_csm_rng_job_id
- *Data type:* uint32_t
- *Value range:* Including 0 up to and including 4294967295
- Default value: -
- *Description:* The job ID for calling the Csm_RandomGenerate() function by the SOK- FM.

[A: SOK_FM_1197]

SOK_use_csm_rng_asynch
- Data type: boolean
- Value range: TRUE, FALSE
- Default value: FALSE
- *Description:* Defines whether the CSM job is used for random number generation (to generate challenges and in the time master of the initial value of the SOK time) asnychron.

[A: SOK_FM_1122]

SOK_broadcast_is_used
- Data type: bool
- Value range: TRUE, FALSE
- Initial value: -
- *Description:* Specifies whether broadcast messages must be protected; if not, all functionalities relevant at SOK time are disabled.

[A: SOK_FM_900]

SOK_ecu_function
- *Data type:* enumeration
- *Value range:* VW_SOK_TIME_SERVER, VW_SOK_PARTICIPANT
- Default value: -
- *Description:* Indicates the role of the participant in the SOK context.
- *Notes:* For ECUs with the role VW_SOK_PARTICIPANT the configuration options with the addition [SOK time server] are not available. Also functional requirements marked with [SOK time server] are not valid for them.

[A: SOK_FM_767]

SOK_time_period
- *Data type:* uint16_t
- *Value range:* Including 1 up to and including 5000
- Default value: 100
- *Description:* The incrementing frequency of the SOK time in milliseconds.

[A: SOK_FM_772]

SOK_time_signal
- *Data type*: string
- Value range: -
- Default value: -

- *Description*: The signal with which the SOK time server transmits the SOK time unprotected.

[A: SOK_FM_773]

[SOK time server]
SOK_num_time_clients

- Data type: uint8_t
- *Value range:* Including 1 up to and including 255
- Default value: -
- *Description:* The number of SOK stations that must be supplied with the authentic SOK time by the SOK time server.

[A: SOK_FM_783]

[SOK time server]
SOK_time_send_period

- Data type: uint8_t
- *Value range:* Including 1 up to and including 255
- Default value: 10
- *Description:* The number of increments of SOK time after which the SOK time server sends an unprotected time message.

[A: SOK_FM_799]

[SOK participant]
SOK_jitter_max

- Data type: uint8_t
- *Value range:* Including 0 up to and including 255
- Default value: 50
- *Description:* The maximum deviation of the internal authenticated SOK time from the reference value of the time master in milliseconds.

[A: SOK_FM_808]

[SOK participant]
SOK_time_request_timeout

- *Data type:* uint16_t
- *Value range:* Including 1 up to and including 65535
- Default value: 250
- *Description:* The maximum time in milliseconds that a SOK subscriber waits for the SOK time server to respond to a request for an authentic SOK time.

[A: SOK_FM_899]

SOK_valid_time_timeout

- *Data type:* uint16_t
- *Value range:* Including 0 up to and including 65535
- Default value: 500
- *Description:* The maximum time in milliseconds that SOK-FM will use a default time value for signature generation or accept during signature verification.

[A: SOK_FM_1192]

SOK_default_time_min_usage

- *Data type:* uint16_t
- *Value range:* Including 0 up to and including 65535
- Default value: 500
- *Description:* The minimum time in milliseconds in which SOK-FM uses a default time value for signature generation. This parameter has no influence on the use of a default time value for signature verification.

[A: SOK_FM_1193]

Only values of SOK_default_time_min_usage that are less than or equal to SOK_valid_time_timeout may be used.

[A: SOK_FM_1123]

SOK_deactivation_development

- Data type: boolean
- Value range: TRUE, FALSE
- Default value: FALSE
- *Description:* Specifies whether signature verification can be disabled via the SOK_function_deactivation diagnostic routine in development mode (persistent and for all SOK PDU IDs) or in serial mode (only for specified SOK PDU IDs and only at runtime).

[A: SOK_FM_1098]

[SOK participant]
SOK_time_development_mode

- *Data type*: boolean
- *Value range*: TRUE, FALSE
- Default value: FALSE
- *Description*: A dummy value is returned as Freshness Value for signature verification for Authentic Broadcast if the SOK time is not available.
- *Note*: The dummy value will cause the signature verification to fail. Develo- pment Mode only works if SecOcIgnoreVerificationResult is enabled at the same time.

[A: SOK_FM_1099]

If SOK_time_development_mode == TRUE is set in the configuration when generating the SOK-FM, the generator must check whether Sec OcIgnoreVerficiationResult is also true. If this is not the case, the generation must be aborted with a meaningful error message.

2.2.1.1 SOK_freshness_value_config_array

[A: SOK_FM_820]

- *Data type:* array of structs
- Value range: -
- Default value: -
- *Description:* This field contains the configuration of the Freshness Values for SOK indexed via FreshnessValueId.
- *Note:* Its length is equal to the number of Freshness Value configured for SecOC.

[A: SOK_FM_822]

SOK_freshness_type

- *Data type:* enumeration
- *Value range:* VW_SOK_FRESHNESS_TIME, VW_SOK_FRESHNESS_TIME_SES-SION_SENDER,
- VW_SOK_FRESHNESS_TIME_SESSION_RECEIVER, VW_SOK_FRESHNESS_CR_CHAL-LENGE, VW_SOK_FRESHNESS_CR_RESPONSE, VW_SOK_FRESHNESS_NONE
- Default value: -
- *Description:* The use of the Freshness Value either in the context of authentic broadcast or challenge response or a session for authentic broadcast or challenge response.
- Note: A session allows one SOK time or challenge to be used for multiple messages (with the same FreshnessValueID). Replay protection is provided here by a counter managed by SOK. In the rest of this specification, this is referred to as Session- based Freshness.

[A: SOK_FM_823]

SOK_freshness_name
- Data type: string
- Value range: -
- Default value: -
- *Description:* The unique name of the challenge response use case.

[A: SOK_FM_825]

SOK_freshness_signal
- Data type: string
- Value range: -
- Default value: -
- *Description:* For entries for which SOK_freshness_type has the value VW_SOK_FRESHNESS_CR_CHALLENGE, VW_SOK_FRESHNESS_CR_CHALLENGE_SESSION, VW_SOK_FRESHNESS_CR_RESPONSE or VW_SOK_FRESHNESS_CR_RESPONSE_SESSION has, the signal with which the challenge is transmitted

[A: SOK_FM_826]

SOK_freshness_value
- *Data type:* reference
- Value range: -
- Default value: -
- *Description:* Reference to the memory area containing the Freshness Value.
- *Note:* All references for which SOK_freshness_type has the value VW_SOK_FRESHNESS_TIME or VW_SOK_FRESHNESS_TIME_SESSION_SENDER or VW_SOK_FRESHNESS_TIME_SESSION_RECEIVER, this reference points to the same memory area that contains the big endian representation of the 56 least significant bit of the value of SOK_time.

[A: SOK_FM_1032]

SOK_freshness_session_counter_length
- *Data type*: uint8_t
- Value range: 0...8
- Default value: 0
- *Description*: Length (in bytes) of the counter used for session-based freshness.

[A: SOK_FM_1028]

SOK_freshness_ignore_replay
- Data type: bool
- Value range: TRUE, FALSE
- Default value: FALSE
- *Description:* No replay checks (e.g. check for duplicate use of message counters) are performed for this Freshness Value

[A: SOK_FM_1124]

SOK_freshness_deactivation_in_series
- Data type: bool
- Value range: TRUE, FALSE
- Default value: FALSE
- *Description:* Defines whether the deactivation of the signature verification for the message belonging to this Freshness ValueId is also possible for the series (cf. SOK_FM_1123).

[A: SOK_FM_863]

[SOK participant] SOK_client_name
- Data type: string

- Value range: -
- Default value: -
- *Description:* The unique designation of the SOK participant.

2.2.1.2 [SOK time server] SOK_time_client_config_array

[A: SOK_FM_859]

- *Data type:* array of structs
- Value range: -
- Default value: -
- *Description:* This field contains the configuration of the SOK time server regarding the SOK participants it must serve with the SOK protocol for authentic time distribution.

[A: SOK_FM_860]

SOK_client_name
- Data type: string
- Value range: -
- Default value: -
- *Description:* The unique designation of the SOK participant.

[A: SOK_FM_861]

SOK_freshness_value_id
- *Data type:* uint16_t
- *Value range:* 0 to 65535 inclusive
- Default value: -
- *Description:* FreshnessValueId of the freshness value used in the challenge response process during authentic time distribution.

[A: SOK_FM_862]

SOK_auth_time_response_signal
- *Data type:* reference
- *Value range:* 0 to 65535 inclusive
- Default value: -
- *Description:* Reference to the signal with which the SOK time server authentically transmits the SOK time to the SOK participant.

2.2.2 Runtime status

[I: SOK_FM_751]

This section describes the global status variables of the SOK-FM as well as their initial values.

[A: SOK_FM_752]

SOK_time
- *Data type:* uint64_t
- *Range of values:* Including 0 up to and including 0xFFFFFFFFFFFFFF
- Initial value: 0
- *Description:* The current SOK time

[A: SOK_FM_753]

SOK_time_is_valid
- Data type: bool
- Value range: TRUE, FALSE
- Initial value: FALSE
- *Description:* Indicates whether there is a valid authenticated SOK time in the SOK-FM.

[A: SOK_FM_754]

SOK_clock_count
- *Data type:* uint16_t
- *Value range:* Including 0 up to and including (SOK_time_period + SOK_main_period - 1) in increments of SOK_main_period.
- Initial value: 0

Description: Time elapsed since the last increment of the SOK time by the SOK-FM in milliseconds.

[A: SOK_FM_762]

[SOK time server]
SOK_time_last_sent
- Data type: uint8_t
- *Range:* Including 0 up to and including SOK_time_send_period
- Initial value: 10
- *Description:* Number of increments of the SOK time by the SOK-FM since the last sending of the unprotected SOK time.

[A: SOK_FM_798]

[SOK participant]
SOK_jitter_max_exceeded
- Data type: bool
- Value range: TRUE, FALSE
- Default value: FALSE
- *Description:* The deviation of the internal authenticated SOK time from the reference value of the SOK time server has exceeded the value of SOK_jitter_max.

[A: SOK_FM_809]

[SOK participant]
SOK_time_request_timer
- *Data type:* uint16_t
- *Value range:* Including 0 up to and including (SOK_time_request_timeout + SOK_main_period) in increments of SOK_main_period.
- Initial value: 65535
- *Description:* Past time since the last request to the SOK time server for an au- thentic SOK time in milliseconds.

[A: SOK_FM_857]

[SOK time server]
SOK_time_client_is_queued_array
- *Data type:* array of bool
- Value range: TRUE, FALSE
- Default value: FALSE
- *Description:* This field contains the runtime states of the requests of the SOK participants after an authentic SOK time; indicates whether there is a request for the participant.

2.2.2.1 [SOK participant] SOK_freshness_value_status_array

[A: SOK_FM_892]

- *Data type:* array of structs
- Value range: TRUE, FALSE
- Default value: FALSE
- *Description:* This field contains the runtime states of the Freshness Values for SOK indexed via FreshnessValueId.

[A: SOK_FM_843]

SOK_freshness_value_is_valid
- Data type: bool
- Value range: TRUE, FALSE
- Default value: FALSE
- *Description:* Indicates whether this Freshness Value is currently valid.

[A: SOK_FM_906]

SOK_freshness_value_is_active
- Data type: bool
- Value range: TRUE, FALSE
- Default value: FALSE
- *Description:* Indicates whether this Freshness Value has already been requested by SecOC.

[A: SOK_FM_898]

SOK_freshness_value_time_since_first_pdu
- *Data type:* uint16_t
- *Value range:* Including 0 up to and including 65535
- Default value: 0
- *Description:* Specifies how much time in milliseconds has passed since the first request by SecOC for this Freshness Value.

[A: SOK_FM_946]

SOK_freshness_value_latest_verified_time
- *Data type:* uint64_t
- *Value range:* Including 0 up to and including 0xFFFFFFFFFFFFFF
- *Default value:* 0xFFFFFFFFFFFFFFFF
- *Description:* Contains the value of the SOK time with which the last successful verification for this Freshness Value was performed.

[A: SOK_FM_947]

SOK_freshness_value_latest_verified_time_buffered
- *Data type:* uint64_t
- *Value range:* Including 0 up to and including 0xFFFFFFFFFFFFFF
- *Default value:* 0xFFFFFFFFFFFFFFFF
- *Description:* Contains the value of the last Freshness Value that was passed to Se- cOC by the SOK-FM.

[A: SOK_FM_948]

SOK_freshness_value_counter_buffered
- Data type: uint8_t
- *Value range:* Including 0 up to and including 254
- *Default value:* 255 (invalid)
- *Description:* Contains the value of the message counter passed from SecOC to SecOC-FM.

[A: SOK_FM_894]

SOK_freshness_value_counter_used
- *Data type:* uint16_t (bit field)
- *Value range:* Including 0 up to and including 0xFFFF
- Default value: 0
- *Description:* Indicates whether a counter value has already been consumed during verification of a signature.

[A: SOK_FM_1033]

SOK_freshness_value_session_counter_buffered

- *Data type*: uint64_t
- *Value range*: Including 1 up to and including 0xFFFFFFFFFFFFFF, limited by the value of SOK_freshness_session_counter_length
- *Default value*: 0 (invalid)
- *Description*: Contains the value of the message counter of the session-based freshness for the current signature verification.
- *Note*: This value is passed from SecOC to the SOK-FM.

[A: SOK_FM_1034]

SOK_freshness_value_session_counter_last_used

- *Data type*: uint64_t
- *Range of values*: Including 0 up to and including 0xFFFFFFFFFFFFFF
- Default value: 0
- *Description*: Contains the last value of the message counter of the session-based freshness with which a signature was last created or successfully verified.

2.3 Signal inputs

[A: SOK_FM_795]

[SOK participant]
SOK_time_signal

- *Data type:* uint8_t [7]
- *Description:* Big endian representation of the 7 least significant bytes of the SOK time (unprotected transmission).

[A: SOK_FM_845]

[SOK participant]
SOK_auth_time_signal_<SOK_client_name>

- *Data type:* uint8_t [7]
- *Description:* Big endian representation of the 7 least significant bytes of the SOK time (protected transmission).

[A: SOK_FM_814]

[SOK Time Server]
For each entry in SOK_time_client_config_array:
SOK_auth_time_challenge_<SOK_client_name>

- *Data type:* uint8_t [8]
- *Description:* Challenge of the SOK participant <SOK_client_name> to the SOK time server.

2.3.1 SokFm_CrReceiveChallenge<SOK_freshness_name>

[A: SOK_FM_834]

- *Data type:* uint8_t [8]
- *Description:* Received Challenge

[A: SOK_FM_835]

For each entry in SOK_freshness_value_config_array for which SOK_freshness_type has the value SOK_FRESHNESS_CR_RESPONSE or SOK_FRESHNESS_CR_RESPONSE_SESSION, the SOK-FM must provide a signal input with the name SokFm_CrReceiveChallenge<SOK_freshness_name>. The corresponding values SOK_freshness_signal and SOK_freshness_value are associated with this.

[A: SOK_FM_836]

The SOK-FM must set the referenced value of the SOK_freshness_reference belonging to the signal to the value of the received signal.

[A: SOK_FM_1035]

If the SOK_freshness_type entry corresponding for SokFm_CrReceiveChallenge<SOK_freshness_name> in the SOK_freshness_value_config_array field has the value VW_SOK_FRESH- NESS_CR_RESPONSE_SESSION, the SOK-FM must set the corresponding value of SOK_freshness_value_session_counter_last_used in the SOK_freshness_value_status_array field to its initial value.

[A: SOK_FM_848]

The SOK-FM must set the value of the corresponding SOK_freshness_value_is_valid in the SOK_freshness_value_status_array field to TRUE.

[A: SOK_FM_837]

The SOK-FM must inform other application about its reception via a callout function or appropriate signals after receiving the signal SokFm_CrReceiveChallenge<SOK_freshness_name>.

[I: SOK_FM_1016]

Thus, other applications are notified that a certain request is present and SecOC is ready to sign the response.

2.4 Signal outputs

[A: SOK_FM_794]

[SOK time server]
SOK_time_signal

- *Data type:* uint8_t [7]
- *Description:* Big endian representation of the 7 least significant bytes of the SOK time (unprotected transmission).

[A: SOK_FM_846]

[SOK Time Server]
For each entry in SOK_time_client_config_array:
SOK_auth_time_signal_<SOK_client_name>

- *Data type:* uint8_t [7]
- *Description:* Big endian representation of the 7 least significant bytes of the SOK time (protected transmission).

[A: SOK_FM_813]

[SOK participant]
SOK_auth_time_challenge_<SOK_client_name>

- *Data type:* uint8_t [8]
- *Description:* Challenge of the SOK participant <SOK_client_name> to the SOK time server.

2.5 Service functions

2.5.1 SokFm_MainFunction()

[A: SOK_FM_784]

- Input values: -
- Output values: -
- *Return values:* VW_SOK_OK, VW_SOK_ERROR_RNG

- *Description:* This is the main function of the SOK-FM and is called cyclically every SOK_main_period milliseconds.

[A: SOK_FM_916]

[SOK Participant]
If SOK_time_is_valid has the value FALSE, the SOK-FM must call the function VW_SOK_incTime-Timeout().

[A: SOK_FM_760]

[SOK Time Server]
If SOK_is_valid has the value FALSE, the SOK-FM must call the function VW_SOK_FM_initTime().

[A: SOK_FM_785]

[SOK Time Server]
If the function VW_SOK_FM_initTime() returns the value VW_SOK_ERROR_RNG, the function must abort and return the value VW_SOK_ERROR_RNG.

[A: SOK_FM_804]

[SOK Participant]
If SOK_time_is_valid has the value FALSE or SOK_jitter_max_exceeded has the value TRUE, the SOK-FM must call the function VW_SOK_FM_requestAuthenticatedTime().

[A: SOK_FM_759]

If SOK_time_is_valid is the value TRUE has, then the SOK-FM must call the function VW_SOK_FM_incClockCount().

[A: SOK_FM_867]

[SOK Time Server]
If the function VW_SOK_FM_incClockCount() returns the value VW_SOK_OK_TIME_INCREMENTED, the SOK-FM must call the function VW_SOK_FM_sendAuthenticTimeResponse().

[A: SOK_FM_781]

[SOK Time Server]
If the function VW_SOK_FM_incClockCount() returns the value VW_SOK_OK_TIME_INCREMENTED, the SOK-FM must increment the value of SOK_time_last_sent by 1.

[A: SOK_FM_782]

[SOK Time Server]
If the value of SOK_time_last_sent has reached or exceeded the value SOK_time_send_period, the SOK-FM must call the function VW_SOK_FM_sendUnauthenticatedTime(). After that the SOK-FM has to set the value of SOK_time_last_sent to 0.

2.5.2 SokFm_Deinit()

[A: SOK_FM_775]

- Input values: -
- Output values: -
- Return values: VW_SOK_OK
- *Description:* This function sets all global status variables to their initial values.

[I: SOK_FM_838]

This function is to be called as soon as the ECU can no longer guarantee that the function W_SOK_FM_main() is called at the specified intervals, e.g. when the ECU goes to sleep. Furthermore the status of all SOK protocols is reset. The function puts the SOK-FM in an inactive state with respect to SOK time until the next time the SokFm_MainFunction() is called. Challenge-

response protocols can still be executed.

2.5.3 SokFm_TriggerCrRequest*()

[A: SOK_FM_818]

- Input values: -
- Output values: -
- *Return values:* VW_SOK_OK, VW_SOK_ERROR_SEND, VW_SOK_ERROR_RNG
- *Description:* This function sends a challenge as part of the SOK Challenge- Response procedure.

[A: SOK_FM_824]

For each entry in SOK_freshness_value_config_array for which SOK_freshness_type has the value SOK_FRESHNESS_CR_CHALLENGE or SOK_FRESHNESS_CR_CHALLENGE_SESSION the SOK-FM must provide a service function with the name SokFm_TriggerCrRequ-est<SOK_freshness_name>. The corresponding values SOK_freshness_signal and SOK_freshness_value are associated with this.

[A: SOK_FM_827]

The SOK-FM must call the external function Csm_RandomGenerate() with the job ID SOK_csm_rng_job_id and request a 64 bit long random number SOK_challenge.

[A: SOK_FM_1199]

When calling the Csm_RandomGenerate() interface according to SOK_FM_827, it must be treated as a synchronous or asynchronous interface according to the SOK_use_csm_rng_asynch confuguration option (cf. SOK_FM_1197).

[A: SOK_FM_828]

If the SOK-FM could not get a random number from the CSM, the function must abort and return the return value VW_SOK_ERROR_RNG.

[A: SOK_FM_1036]

If the SOK_freshness_type entry corresponding for SokFm_TriggerCrRequest<SOK_freshness_name> in the SOK_freshness_value_config_array field has the value VW_SOK_FRESH- NESS_CR_CHALLENGE_SESSION, SOK-FM must set the corresponding value of SOK_freshness_value_session_counter_last_used in the SOK_freshness_value_status_array field to its initial value.

[A: SOK_FM_830]

The SOK-FM must transmit the value of SOK_challenge to the AUTOSAR ComStack as signal SOK_freshness_signal of the corresponding service function.

[A: SOK_FM_831]

If the transfer was not successful, the function must abort and return the value VW_SOK_ERROR_SEND.

[A: SOK_FM_829]

The SOK-FM must set the referenced value of the SOK_freshness_reference belonging to the corresponding service function to the value SOK_challenge.

[A: SOK_FM_851]

The SOK-FM must set the corresponding value of SOK_freshness_value_is_valid in the SOK_freshness_status_array field to TRUE.

[A: SOK_FM_832]

The function returns VW_SOK_OK.

2.5.4 SokFm_SecOC_VerificationStatusCallout()

[A: SOK_FM_903]

- *Input values:* SecOCFreshnessValueID, SecOCDataId, VerificationStatus

- Output values: -
- Return values: -
- *Description:* This function receives the result of a signature verification from the SecOC module.

[I: SOK_FM_921]

This function is called by SecOC after the verification of an incoming message has been completed.

[A: SOK_FM_957]

If VerificationStatus = SECOC_VERIFICATIONFAILURE, VerificationStatus = SECOC_NO_VERIFICATION, VerificationStatus = SECOC_VERIFICATIONFAILURE_OVERRIDEN, VerificationStatus = SECOC_AUTHENTICATIONBUILDFAILURE or VerificationStatus = SECOC_FRESHNESSFAILURE, the function terminates.

[A: SOK_FM_958]

In the following, VerificationStatus = SECOC_VERIFICATIONSUCCESS.

[A: SOK_FM_959]

If for SecOCFreshnessValueID the corresponding entry SOK_freshness_type in the field SOK_freshness_config_array has the value VW_SOK_FRESHNESS_CR_CHALLENGE, the SOK-FM sets the corresponding entry SOK_freshness_value_is_valid in the field SOK_freshness_sta- tus_array to FALSE. After that the function terminates.

[A: SOK_FM_1037]

If for SecOCFreshnessValueID the corresponding entry SOK_freshness_type in the field SOK_freshness_config_array has the value VW_SOK_FRESHNESS_CR_CHALLENGE_SESSION, the SOK-FM sets the corresponding entry SOK_freshness_value_is_valid in the field SOK_freshness_status_array to FALSE and the value of SOK_freshness_value_session_coun- ter_last_used to SOK_freshness_value_session_counter_buffered. After that the function terminates.

[A: SOK_FM_960]

In the following it is valid that for SecOCFreshnessValueID the corresponding entry SOK_freshness_type in the field SOK_freshness_config_array has the value VW_SOK_FRESHNESS_TIME or VW_SOK_FRESHNESS_TIME_SESSION_RECEIVER.

[A: SOK_FM_1038]

If for SecOCFreshnessValueID the corresponding SOK_freshness_type entry in the SOK_freshness_config_array field has the value VW_SOK_FRESHNESS_TIME and SOK_freshness_value_latest_verified_time_buffered > SOK_freshness_value_latest_verified_time, the SOK-FM sets the value of SOK_freshness_value_counter_used to its initial value.

[A: SOK_FM_961]

If SOK_freshness_value_latest_verified_time_buffered > SOK_freshness_value_latest_verified_time, the SOK-FM sets the value of SOK_freshness_value_latest_verified_time to the value of SOK_freshness_value_latest_time_buffered.

[A: SOK_FM_962]

If for SecOCFreshnessValueID the corresponding SOK_freshness_type entry in the SOK_freshness_config_array field has the value VW_SOK_FRESHNESS_TIME, the SOK-FM sets the bit number <SOK_freshness_value_coun- ter_buffered> to 1 in the value of SOK_freshness_value_counter_used.

[A: SOK_FM_1039]

If for SecOCFreshnessValueID the corresponding SOK_freshness_type entry in the SOK_freshness_config_array field has the value VW_SOK_FRESHNESS_TIME_SESSION_RECEIVER, the SOK-FM sets the corresponding value of SOK_freshness_value_session_coun- ter_last_used to SOK_freshness_value_session_counter_buffered.

2.5.5 SokFm_Init()

[A: SOK_FM_1185]

- Input values: -
- Output values: -
- Return values: -
- *Description:* This function initializes the SOK-FM module

[A: SOK_FM_1186]

When calling this function SOK-FM must initialize the runtime variables.

[A: SOK_FM_1187]

Subsequently, the availability of the VKMS keys used by SOK must be queried according to the description in the section "Integration against VKMS" when using VKMS.

[A: SOK_FM_1188]

If the configuration value SOK_deactivation_development == TRUE, SOK-FM must read the value SOK_function persisted according to SOK_FM_1174 from the NvM.

[A: SOK_FM_1189]

If the value SOK_function == 0x01 read out according to SOK_FM_1188, SOK-FM must call VW_SOK_FM_deactivate_SOK().

2.6 Callback functions

2.6.1 SecOC_GetTxFreshness()

[A: SOK_FM_840]

- *Input values:* SecOCFreshnessValueID, SecOCFreshnessValueLength
- *Output values:* SecOCFreshnessValue
- *Return values:* Std_ReturnType (E_OK, E_NOT_OK)
- *Description:* This function provides the SecOC module with a Freshness Value to create an outgoing signature.

[I: SOK_FM_888]

The function is called by the SecOC module as soon as a freshness value is needed to generate the signature of an outgoing message.

[I: SOK_FM_934]

This function returns a Freshness Value for each valid FreshnessValueID if the parameters of the request were correct. If there is no valid Freshness Value at the time of the request (e.g. no SOK time available or no new challenge set), default values are returned, which will not lead to successful verification on the receiver side. The SOK-FM thus never prevents the sending of a message to be protected if it has been integrated correctly. Furthermore, the sent data can be used to draw conclusions about the current state of the sender, which improves testability.

[A: SOK_FM_841]

If there is no entry in the SOK_freshness_va- lue_config_array field for the SecOCFreshnessValueID passed, the function terminates and returns E_NOT_OK.

[A: SOK_FM_881]

If for the transferred SecOCFreshnessValueID the corresponding entry SOK_fresh- ness_type in the field SOK_freshness_value_config_array has a value other than VW_SOK_FRESH-NESS_TIME, VW_SOK_FRESHNESS_TIME_SESSION_SENDER, VW_SOK_FRESH-NESS_CR_RESPONSE or VW_SOK_FRESHNESS_CR_RESPONSE_SESSION, the function terminates and returns the value E_NOT_OK.

[A: SOK_FM_882]

If the corresponding entry SOK_freshness_type in the field SOK_freshness_value_config_array contains the value VW_SOK_FRESHNESS_TIME or VW_SOK_FRESHNESS_TIME_SESSION_SEN-
DER, the SOK-FM sets the output value SecOCFreshnessValue to the output value SOK_freshness_value_ret of the function VW_SOK_FM_getTxTime().

[A: SOK_FM_887]

If for the transferred SecOCFreshnessValueID the corresponding entry SOK_fresh- ness_type in the field SOK_freshness_value_config_array has the value VW_SOK_FRESH-NESS_CR_RESPONSE or VW_SOK_FRESHNESS_CR_RESPONSE_SESSION, the SOK-FM sets the output value SecOCFreshnessValue to the output value SOK_freshness_va- lue_ret of the function VW_SOK_FM_getTxChallenge().

[A: SOK_FM_904]

If the transferred value of SecOCFreshnessValueLength does not correspond to the length in bit of the output value of VW_SOK_FM_getTxTime or VW_SOK_FM_getTxChallenge, the function terminates and returns E_NOT_OK.

[A: SOK_FM_1020]

The SOK-FM starts a timer for this Freshness Value with initial value 0. The

[A: SOK_FM_889]

function returns E_OK.

2.6.2 SecOC_SPduTxConfirmation()

[A: SOK_FM_1022]

- *Input values:* SecOCFreshnessValueID
- Output values: -
- Return values: -
- *Description:* This function is used by the SecOC module to inform the SOK-FM about the successful transfer of a Secured I-PDU to the AUTOSAR communication stack.

[I: SOK_FM_1023]

This function is used as an indirect confirmation of successful signature creation.

[A: SOK_FM_1024]

The SOK-FM stops the timer of the timer belonging to SecOCFreshnessValueID (see SOK_FM_1020).

2.6.3 SecOC_GetRxFreshnessAuthData()

[A: SOK_FM_924]

- *Input values:* SecOCFreshnessValueID, SecOCFreshnessValueLength, SecOCTruncated-FreshnessValue, SecOCTruncatedFreshnessValueLength, SecOCAuthVerifyAttempts, SecO-CAuthDataFreshnessValue
- *Output values:* SecOCFreshnessValue
- *Return values:* Std_ReturnType (E_OK, E_NOT_OK)
- *Description:* This function provides the SecOC module with a Freshness Value to validate an incoming signature.

[I: SOK_FM_925]

The function is called by the SecOC module as soon as a freshness value is needed to verify the signature of an incoming message.

[A: SOK_FM_926]

If there is no entry in the SOK_freshness_va- lue_config_array field for the SecOCFreshnessValueID passed, the function terminates and returns E_NOT_OK.

[A: SOK_FM_928]

If the corresponding entry SOK_fresh- ness_type in the field SOK_freshness_value_config_array has a different value than VW_SOK_FRESH- NESS_TIME, VW_SOK_FRESHNESS_TIME_SESSION_RECEIVER, VW_SOK_FRESH- NESS_CR_CHALLENGE or VW_SOK_FRESHNESS_CR_CHALLENGE_SESSION for the transferred SecOCFreshnessValueID, the following error occurs
aborts the function and returns the value E_NOT_OK.

[A: SOK_FM_929]

If the corresponding entry SOK_fresh- ness_type in the field SOK_freshness_value_config_array has the value VW_SOK_FRESH- NESS_CR_CHALLENGE or VW_SOK_FRESHNESS_CR_CHALLENGE_SESSION for the transferred SecOCFreshnessValueID, the following is set
the SOK-FM sets the output value SecOCFreshnessValue to the output value SOK_fresh- ness_value_ret of the function VW_SOK_FM_getRxChallenge() and returns its return value.

[A: SOK_FM_935]

If the corresponding entry SOK_freshness_type in the field SOK_freshness_value_config_array contains the value VW_SOK_FRESHNESS_TIME or VW_SOK_FRESHNESS_TIME_SESSION_RECEI-
VER, the SOK-FM sets the output value SecOCFreshnessValue to the output value SOK_freshness_value_ret of the function VW_SOK_FM_getRxTime() and returns its return value.

[A: SOK_FM_927]

If the transferred value of SecOCFreshnessValueLength does not correspond to the length in bit of the output value of VW_SOK_FM_getRxTime or VW_SOK_FM_getRxChallenge, the function terminates and returns E_NOT_OK.

2.7 Internal interfaces

2.7.1 [SOK time server] VW_SOK_FM_initTime()

[A: SOK_FM_776]

- Input values: -
- Output values: -
- *Return values:* VW_SOK_OK, VW_SOK_ERROR_RNG

Description: This function initializes the value of SOK_time with a random number

[A: SOK_FM_763]

For this the SOK-FM calls the external function Csm_RandomGenerate() SOK_csm_rng_job_id and requests a 64 bit long random number SOK_time_random.

[A: SOK_FM_1200]

When calling the Csm_RandomGenerate() interface according to SOK_FM_763, it must be treated as a synchronous or asynchronous interface according to the SOK_use_csm_rng_asynch confuguration option (cf. SOK_FM_1197).

[A: SOK_FM_777]

If the SOK-FM could not get a random number from the CSM, the function must abort and return the return value VW_SOK_ERROR_RNG.

[A: SOK_FM_768]

The SOK-FM sets the value of SOK_time to the value of SOK_time_random.

[A: SOK_FM_764]

After that the SOK-FM sets the 9 most significant bits of SOK_time to 0.

[A: SOK_FM_791]

SOK_time thus has an initial value in the range from 0 up to and including 0x007FFFFFFFF.

[A: SOK_FM_769]

Afterwards the SOK-FM sets the value of SOK_time_is_valid to TRUE.

[A: SOK_FM_778]

After that the function VW_SOK_OK returns.

2.7.2 VW_SOK_FM_incClockCount()

[A: SOK_FM_780]

- Input values: -
- Output values: -
- *Return values:* VW_SOK_OK, VW_SOK_OK_TIME_INCREMENTED
- *Description:* This function increments the value of SOK_clock_count by SOK_main_pe- riod.

[A: SOK_FM_758]

If SOK_clock_count reaches or exceeds SOK_time_period, the function must increment the value of SOK_time by one and set the value of SOK_clock_count to the value (SOK_clock_count modulo SOK_time_period).

[A: SOK_FM_774]

[SOK Time Server]
If the value of SOK_time was incremented, the function returns the value VW_SOK_TIME_IN- CREMENTED. Otherwise the function returns the value VW_SOK_OK.

[A: SOK_FM_1040]

[SOK Participant]
If the value of SOK_time has been incremented, the SOK-FM sets the corresponding value SOK_freshness_value_ses- sion_counter_last_used to its initial value for each entry in SOK_freshness_value_config_array for which SOK_freshness_type has the value SOK_FRESH- NESS_TIME_SESSION_SENDER.

2.7.3 [SOK participant] VW_SOK_FM_incTimeTimeout()

[A: SOK_FM_918]

- Input values: -
- Output values: -
- Return values: -
- *Description:* This function increments the status of the Freshness Value with respect to the past time in which default values can be used.

[A: SOK_FM_919]

For each Freshness Value, for which the value VW_SOK_FRESHNESS_TIME or VW_SOK_FRESH- NESS_TIME_SESSION_SENDER or VW_SOK_FRESHNESS_TIME_SESSION_RECEIVER is entered in the field SOK_freshness_value_config_array for the entry SOK_freshness_type.
and in the SOK_freshness_value_status_array the entry SOK_freshhness_value_is_active has the value TRUE, the SOK-FM increments the corresponding value of SOK_freshness_value_time_since_first_pdu in the SOK_freshness_value_status array by the value of SOK_main_period.

[A: SOK_FM_920]

If such an incremented value exceeds the SOK_valid_time_timeout value, the SOK- FM must set this value to the value of SOK_valid_time_timeout.

2.7.4 [SOK participant] VW_SOK_FM_receiveUnauthenticatedTime()

[A: SOK_FM_797]

- *Input values:* SOK_time_signal
- Output values: -
- Return values: -
- *Description:* This function of the SOK-FM is called as soon as the SOK station receives the signal SOK_time_signal. It calculates the deviation of the internal SOK time from the received reference value.

[A: SOK_FM_803]

If SOK_time_is_valid has the value FALSE, the function terminates.

[A: SOK_FM_800]

The SOK-FM must convert the received signal SOK_time_signal (56 bits big endian) to the uint64_t value SOK_unauthenticated_time.

[A: SOK_FM_801]

The SOK-FM must calculate the deviation of the internal authenticated SOK time from the received unprotected SOK time:
SOK_jitter = SOK_time_period * (SOK_unauthenticated_time - SOK_time) - SOK_clock_count

[A: SOK_FM_802]

If the absolute value of SOK_jitter exceeds the value of SOK_jitter_max, the SOK-FM must set the value of SOK_jitter_max_exceeded to TRUE.

2.7.5 [SOK participant] VW_SOK_FM_requestAuthenticatedTime()

[A: SOK_FM_806]

- Input values: -
- Output values: -
- Return values: -
- *Description:* This function sends a request for an authentic SOK time to the SOK time server.

[A: SOK_FM_807]

If SOK_time_request_timer has reached or exceeded the value SOK_time_request_timeout, the SOK-FM must call the service function SokFm_TriggerCrRequestSecuredSOKTime().

[A: SOK_FM_812]

If the function SokFm_TriggerCrRequestSecuredSOKTime() returns the value VW_SOK_OK, the SOK-FM must set the value of SOK_time_request_timer to the value 0 and then abort.

[A: SOK_FM_811]

If SOK_time_request_timer is smaller than the value SOK_time_request_timeout, the SOK- FM must increment the value of SOK_time_request_timer by SOK_main_period.

2.7.6 [SOK participant] VW_SOK_FM_receiveAuthenticatedTime()

[A: SOK_FM_875]

- *Input values:* SOK_auth_time_signal_<SOK_client_name>

- Output values: -
- Return values: -
- *Description:* This function is called called as soon as the SOK-FM receives the signal SOK_auth_time_signal_<SOK_client_name> is received.

[A: SOK_FM_877]

The SOK-FM must convert the received signal SOK_auth_time_signal_<SOK_client_name> (56 bits big endian) to the uint64_t value SOK_authenticated_time.

[A: SOK_FM_876]

The SOK-FM sets the value of SOK_time to the value of SOK_authenticated_time.

[A: SOK_FM_878]

The SOK-FM sets the value of SOK_time_is_valid to TRUE.

[A: SOK_FM_886]

For each entry SOK_freshness_type in the field SOK_freshness_config_array with the value VW_SOK_FRESHNESS_TIME or VW_SOK_FRESHNESS_TIME_SESSION_SENDER oider VW_SOK_FRESHNESS_TIME_SESSION_RECEIVER the SOK-FM sets the corresponding value of SOK_freshness_value_is_valid in the field SOK_freshness_value_status_array to TRUE.

[A: SOK_FM_879]

The SOK-FM sets the following values to their initial values:
- SOK_clock_count
- SOK_jitter_max_exceeded
- SOK_time_request_timer

2.7.7 [SOK participant] VW_SOK_FM_getTxTime()

[A: SOK_FM_891]

- *Input values:* SecOCFreshnessValueID
- *Output values:* SOK_freshness_value_ret (uint8_t [n])
- Return values: -
- *Description:* This function provides the freshness value constructed from the current SOK time and, if session-based freshness is used, also from the counter.
- *Note*: The length n of the output value results as 8 + value SOK_freshness_session_counter_length from the SecOCFreshnessValueId corresponding element of the SOK_freshness_value_config_array field.

[A: SOK_FM_907]

If the SecOCFreshnessValueID corresponding value SOK_freshness_value_time_since_first_pdu in the SOK_freshness_value_status_array field is less than the value of SOK_default_time_min_usage, the SOK-FM sets the value of SOK_freshness_value_ret to the value 0xFFFFFFFFFF (big endian).

[A: SOK_FM_1195]

If the SecOCFreshnessValueID corresponding value SOK_freshness_value_time_since_first_pdu in the SOK_freshness_value_status_array field is less than the value of SOK_default_time_timeout and SOK_time_is_valid has the value FALSE, the SOK-FM sets the value of SOK_freshness_value_ret to the value 0xFFFFFFFFFF (big endian).

[A: SOK_FM_895]

If the SecOCFreshnessValueID corresponding value SOK_freshness_value_time_since_first_pdu in the SOK_freshness_value_status_array field is greater than or equal to the value of SOK_default_time_min_usage and SOK_time_is_valid has the value TRUE, the SOK- FM sets the value of SOK_freshness_value_ret to the 64 bit big endian representation of the value of SOK_time.

[A: SOK_FM_908]

If the SecOCFreshnessValueID corresponding value SOK_freshness_value_time_since_first_pdu in the SOK_freshness_value_status_array field is greater than or equal to the value of SOK_valid_time_timeout and SOK_time_is_valid has the value FALSE, the SOK-FM sets the value of SOK_freshness_value_ret to the value 0xF00000000000 (big endian).

[A: SOK_FM_1041]

If the SecOCFreshnessValueID corresponding value SOK_freshness_type in the field SOK_freshness_value_config_array has the value VW_SOK_FRESHNESS_TIME_SESSION_SENDER, the SOK-FM in the cases SOK_FM_907, SOK_FM_895 and SOK_FM_908 the SOK-FM then increments the value of SOK_freshness_value_session_counter_last_used by 1 and sets SOK_freshness_value_ret to the concatenation of the value SOK_freshness_vaule_ret with the value SOK_freshness_value_session_counter_last_used. The concatenation must be such that the SOK_freshness_value_session_counter_last_used value is at the least significant bit.

[A: SOK_FM_909]

The SOK-FM sets the corresponding SOK_freshness_value_is_active value in the SOK_freshness_value_status_array field to TRUE.

2.7.8 [SOK participant] VW_SOK_FM_getRxTime()

[A: SOK_FM_937]

- *Input values:* SecOCFreshnessValueID, SecOCAuthVerifyAttempts, SecOCAuthDataFreshnessValue, SecOCTruncatedFreshnessValue
- *Output values:* SOK_freshness_value_ret (uint8_t [n])
- *Return values:* Std_ReturnType (E_OK, E_NOT_OK)
- *Description:* This function provides the freshness value constructed from the current SOK time and, if session-based freshness is used, also from the counter.
- *Note*: The length n of the output value results as 8 + value SOK_freshness_session_counter_length from the SecOCFreshnessValueId corresponding element of the SOK_freshness_value_config_array field.

[A: SOK_FM_938]

The output value of this function depends on the following circumstances:

- Is there a valid SOK time (SOK_time_is_valid)?
- How much time has passed since a Freshness Value was first requested for this Freshness Value ID after initialization of the SOK-FM (associated value of SOK_freshness_value_time_since_first_pdu in SOK_freshness_value_status_array)?
- Is this time within SOK_valid_time_timeout?
- Is the time development mode active (SOK_time_development_mode, see SOK_FM_1098)?
- Has a successful verification already been performed for this Freshness Value ID with the valid SOK time(associated value of SOK_freshness_value_is_valid in SOK_freshness_value_status_array)?
- What was the value of the last SOK time with which a successful verification was performed (SOK_freshness_value_latest_verified_time)?
- Is the replay protection for this freshness value disabled (SOK_freshness_value_ignore_replay)?
- Which message counters have already been consumed by a successful verification for a Freshness Value (SOK_freshness_value_counter_used)?
- If the message counter of the session-based freshness is valid (SecOCTruncatedFreshnessValue > SOK_freshness_value_session_counter_last_used)?

[I: SOK_FM_939]

The SecOC module can make a maximum of 4 requests for a freshness value for a received protected message. SecOCAuthVerifyAttempts specifies how many requests are made in a row.

[A: SOK_FM_940]

The following values are possible candidates as output values of the function:

- SOK_time_current: The 64 bit big endian representation of the value of SOK_time.
- SOK_time_previous: The 64 bit big endian representation of the value of SOK_time minus 1.
- SOK_time_next: The 64 bit big endian representation of the value of SOK_time plus 1.
- SOK_time_default: The big endian value 0xFF FF FF FF FF FF FF FF FF FF
- SOK_time_dev: The big endian value 0xFF FF FF FF FF FF F0

[A: SOK_FM_941]

The SOK-FM creates a list SOK_freshness_value_candidate_list of candidates depending on the values specified in SOK_FM_938. The order of the elements of the list corresponds to the order in which they are added to the list.

[A: SOK_FM_953]

If SecOCAuthDataFreshnessValue has length 0, SecOCAuthDataFreshnessValue is interpreted as value 0.

[A: SOK_FM_949]

Pseudocode macro definition _fv_is_eligible(FV):
{
FV < SOK_freshness_value_latest_verified_time -> FALSE;
FV > SOK_freshness_value_latest_verified_time -> TRUE;
/* For the case FV = SOK_freshness_value_latest_verified_time and SOK_freshness_type = VW_SOK_FRESHNESS_TIME */
SOK_freshness_value_ignore_replay = TRUE? -> TRUE;
Bit in SOK_freshness_value_counter_used at position <SecOCAuthDataFreshnessValue> is set -> FALSE;
Bit in SOK_freshness_value_counter_used at position <SecOCAuthDataFreshnessValue> is not set -> TRUE;
/* For the case FV = SOK_freshness_value_latest_verified_time and SOK_freshness_type = VW_SOK_FRESHNESS_TIME_SESSION_RECEIVER */
SecOCTruncatedFreshnessValue > SOK_freshness_value_session_counter_last_used -> TRUE;
SecOCTruncatedFreshnessValue <= SOK_freshness_value_session_counter_last_used -> FALSE
}

[A: SOK_FM_952]

If SOK_freshness_value_is_valid = FALSE and SOK_freshness_value_time_since_first_pdu < SOK_valid_time_timeout:
SOK_time_default is added to SOK_freshness_value_candidate_list.

[A: SOK_FM_1105]

If SOK_freshness_value_is_valid = FALSE, SOK_freshness_value_time_since_first_pdu >= SOK_valid_time_timeout and SOK_time_development_mode = TRUE:
SOK_time_dev is added to SOK_freshness_value_candidate_list.

[A: SOK_FM_944]

If SOK_time_is_valid = TRUE and _fv_is_eligible(SOK_time_current) = TRUE:
SOK_time_current is added to SOK_freshness_value_candidate_list.

[A: SOK_FM_945]

If SOK_time_is_valid = TRUE and _fv_is_eligible(SOK_time_previous) = TRUE:
SOK_time_previous is added to SOK_freshness_value_candidate_list.

[A: SOK_FM_942]

If SOK_time_is_valid = TRUE and _fv_is_eligible(SOK_time_next) = TRUE:
SOK_time_next is added to SOK_freshness_value_candidate_list.

[A: SOK_FM_950]

If SOK_freshness_value_candidate_list contains less than <SecOCAuthVerifyAttempts> + 1 candi- dates, the function returns E_NOT_OK.

[A: SOK_FM_951]

The SOK-FM sets the value of SOK_freshness_value_ret to the value of list element with index<SecOCAuthVerifyAttempts> of the list SOK_freshness_value_cadidate_list. The first list element has index 0.

[A: SOK_FM_954]

The SOK-FM sets the value of SOK_freshness_value_latest_verified_time_buffered to the value of SOK_freshness_value_ret.

[A: SOK_FM_1042]

If the SecOCFreshnessValueID corresponding value SOK_freshness_type in the field SOK_freshness_value_config_array has the value VW_SOK_FRESHNESS_TIME_SESSION_RECEIVER, the SOK-FM then sets SOK_reshness_value_ret to the concatenation of the value SOK_freshness_value_ret with the value SecOCTruncatedFreshnessValue. The concatenation must be done in such a way that the SecOCTruncatedFreshnessValue value is at the least significant bit.

[A: SOK_FM_1043]

If the SecOCFreshnessValueID corresponding value SOK_freshness_type in the SOK_freshness_value_config_array has the value VW_SOK_FRESHNESS_TIME_SESSION_RECEIVER, the SOK_FM sets the SOK_freshness_value_session_counter_buffered value to the value of SecOCTruncatedFreshnessValue.

[A: SOK_FM_955]

If the SecOCFreshnessValueID corresponding value SOK_freshness_type in the SOK_freshness_value_config_array field has the value VW_SOK_FRESHNESS_TIME, the SOK-FM sets the value of SOK_freshness_value_counter_buffered to the value of SecOCAuthDataFreshnessValue.

[A: SOK_FM_956]

The function returns E_OK.

2.7.9 VW_SOK_FM_getTxChallenge()

[A: SOK_FM_912]

- *Input values:* SecOCFreshnessValueID
- *Output values:* SOK_freshness_value_ret (uint8_t [n])
- Return values: -
- *Description:* This function provides the Freshness Value constructed from a Challenge and, when using the Ses- sion-based Freshness, additionally from the counter.
- *Note*: The length n of the output value results as 8 + value SOK_freshness_session_counter_length from the SecOCFreshnessValueID corresponding element of the SOK_freshness_value_config_array field.

[A: SOK_FM_914]

If the SecOCFreshnessValueID corresponding value SOK_freshness_value_is_valid in the SOK_freshness_value_status_array field has the value TRUE, the SOK-FM sets the value of SOK_freshness_value_ret to the corresponding value of SOK_freshness_value in the SOK_freshness_value_config_array field.

[A: SOK_FM_913]

If the SecOCFreshnessValueID corresponding value SOK_freshness_value_is_valid in the SOK_freshness_value_status_array field has the value FALSE, the SOK-FM sets the value of SOK_freshness_value_ret to the value 0xF00000000000 (big endian).

[A: SOK_FM_1044]

If the SecOCFreshnessValueID corresponding value SOK_freshness_type in the field SOK_freshness_value_config_array has the value VW_SOK_FRESHNESS_CR_RESPONSE_SENDER, in the case of SOK_FM_913 and SOK_FM_914, the SOK-FM then increments the value of SOK_freshness_value_session_counter_last_used by 1 and then sets SOK_freshness_value_ret to concatenate the value SOK_freshness_vaule_ret with the value SOK_freshness_value_session_counter_last_used. The concatenation must be such that the SOK_freshness_value_session_counter value is at the least significant bit.

[A: SOK_FM_915]

The SOK-FM sets the corresponding value SOK_freshness_value_is_valid in the field SOK_freshness_value_status_array to the value FALSE.

2.7.10 VW_SOK_FM_getRxChallenge()

[A: SOK_FM_931]

- *Input values:* SecOCFreshnessValueID, SecOCTruncatedFreshnessValue
- *Output values:* SOK_freshness_value_ret (uint8_t [n])
- *Return values:* Std_ReturnType (E_OK, E_NOT_OK)
- *Description:* This function provides the Freshness Value constructed from a Challenge and, when using the Ses- sion-based Freshness, additionally from the counter.
- *Note*: The length n of the output value results as 8 + value SOK_freshness_session_counter_length from the SecOCFreshnessValueID corresponding element of the SOK_freshness_value_config_array field.

[A: SOK_FM_1045]

If the SecOCFreshnessValueID corresponding value SOK_freshness_type in the SOK_freshness_value_config_array field has the value VW_SOK_FRESHNESS_CR_CHALLENGE_SESSION and the value of SecOCTruncatedFreshnessValue is less than or equal to the SecOCFreshnessValueID corresponding value of SOK_freshness_value_session_counter_last_used in the SOK_freshness_value_status_array field, the function E_NOT_OK returns.

[A: SOK_FM_932]

If the SecOCFreshnessValueID corresponding value SOK_freshness_value_is_valid in the SOK_freshness_value_status_array field has the value TRUE, the SOK-FM sets the value of SOK_freshness_value_ret to the corresponding value of SOK_freshness_value in the SOK_freshness_value_config_array field.

[A: SOK_FM_1046]

If the SecOCFreshnessValueID corresponding value SOK_freshness_type in the SOK_freshness_value_config_array has the value VW_SOK_FRESHNESS_CR_CHALLENGE_SESSION and the value of SecOCTruncatedFreshnessValue is greater than the corresponding value of SOK_fresh- ness_value_session_counter_last_used in the SOK_freshness_value_status_array, the SOK-FM then sets SOK_freshness_value_ret to concatenate the SOK_fresh- ness_value_ret value with the SecOCTruncatedFreshnessValue value. The concatenation must be done in such a way that the SecOCTruncatedFreshnessValue value is at the least significant bit.

[A: SOK_FM_1047]

If the SecOCFreshnessValueID corresponding value SOK_freshness_type in the field SOK_freshness_value_config_array has the value VW_SOK_FRESHNESS_CR_CHALLENGE_SESSION

and

the value of SecOCTruncatedFreshnessValue is greater than the corresponding value of SOK_freshness_value_session_counter_last_used in the SOK_freshness_value_status_array field, the SOK_FM subsequently sets SOK_freshness_value_session_counter_buffered to the value of SecOCTruncatedFreshnessValue.

[A: SOK_FM_1048]

After SOK_FM_932 or SOK_FM_1047 the function returns E_OK.

[A: SOK_FM_933]

If the SecOCFreshnessValueID corresponding value SOK_freshness_value_is_valid in the SOK_freshness_value_status_array field has the value FALSE, the function returns E_NOT_OK.

2.7.11 [SOK time server] VW_SOK_FM_sendUnauthenticatedTime()

[A: SOK_FM_787]

- Input values: -
- Output values: -
- *Return values:* VW_SOK_OK, VW_SOK_ERROR_SEND
- *Description:* This function of the SOK time server sends an unprotected message with the current SOK time to all SOK participants.

[A: SOK_FM_789]

The SOK-FM must transmit the current value of SOK_time to the AUTOSAR ComStack as signal SOK_time_signal.

[A: SOK_FM_790]

If the transfer was successful, the function must return the value VW_SOK_OK. If the transfer was not successful, the function must return the value VW_SOK_ERROR_SEND.

2.7.12 [SOK time server] VW_SOK_FM_queueAuthenticTimeResponse*()

[A: SOK_FM_854]

- *Input values:* SOK_auth_time_challenge_<SOK_client_name>
- Output values: -
- Return values: -
- *Description:* This function is called when the SOK time server receives the signal SOK_auth_time_challenge_<SOK_client_name>.

[A: SOK_FM_855]

For each SOK participant from SOK_time_client_config_array the SOK time server must provide a func- tion VW_SOK_FM_queueAuthenticTimeResponse<SOK_client_name>(), which is called as soon as it receives a request for authentic time distribution from the corresponding participant.

[A: SOK_FM_864]

The request is made in the form of the signal associated SOK_auth_time_challenge_<SOK_client_name> of the SOK subscriber. The configuration of the signal associated with the subscriber is done in the SOK_freshness_signal entry in the SOK_freshness_value_config_array field and is referenced via SOK_freshness_value_id in the SOK_time_client_config_array field.

[A: SOK_FM_865]

The SOK time master must call the associated service function VW_SOK_FM_setChallenge<SOK_freshness_name>() and pass the value of SOK_auth_time_challenge_<SOK_client_name> as input parameter.

[A: SOK_FM_866]

The SOK time master must set the associated value in the SOK_time_client_is_queued_array field to TRUE.

2.7.13 [SOK time server] VW_SOK_FM_sendAuthenticTimeResponse()

[A: SOK_FM_869]

- Input values: -
- Output values: -
- Return values: -
- *Description:* This function is called as soon as the SOK time is incremented by 1 in the SOK time server and sends an authentic SOK time to the requesting SOK participants.

[A: SOK_FM_870]

The SOK time server must send an authentic SOK time to the corresponding subscriber for each entry in the SOK_time_client_is_queued_array field that has the value TRUE.

[A: SOK_FM_871]

For this purpose, the SOK time master closes from the entries in the SOK_time_client_is_queued_array field with the values TRUE to the assigned signals SOK_auth_time_response_signal in the SOK_time_client_config_array field.

[A: SOK_FM_872]

The SOK time master sends the current value of SOK_time in each corresponding signal SOK_auth_time_response_signal.

[A: SOK_FM_873]

For each successfully sent signal, the SOK time server sets the value of the corresponding entry in the SOK_time_client_is_queued_array field to FALSE.

2.7.14 VW_SOK_FM_deactivate_SOK.

[A: SOK_FM_1126]

- Input values: -
- Output values: -
- *Return values:* E_OK, 0x30 = VW_SOK_ERROR_DEACTIVATING_VERIFICATION_FAILED, 0x32 = VW_SOK_ERROR_ACTIVATING_DEFAULT_AUTHENTICATION_FAILED, 0x 34 = VW_SOK_ERROR_MULTIPLE_FAILURES.
- *Description:* This function is called to perform signature verification override using the SecOc_VerifyStatusOverride interface.

[A: SOK_FM_1130]

When this function is called, the SOK-FM must first compile the list of FreshnessValueIds for which the signature check is to be switched off.

[A: SOK_FM_1131]

If SOK_deactivation_development == TRUE, the list from SOK_FM_1127 consists of all FreshnessValueIds received from this controller (i.e. the FreshnessValu- eIds for each SecOcRxPduProcessing container belonging to a FreshnessValueId of an entry from SOK_freshness_config_array).

[A: SOK_FM_1132]

If SOK_deactivation_development == FALSE, the list of SOK_FM_1127 consists of all FreshnessValueIds received from this controller and for which SOK_fresh-

ness_deactivation_in_series == TRUE (i.e. from the FreshnessValueIds for each SecOcRxPdu-Processing container belonging to a FreshnessValueId of an entry in SOK_freshness_config_array for which SOK_freshness_deactivation_in_series == TRUE).

[A: SOK_FM_1133]

Subsequently, the SOK-FM must call the SecOc_VerifyStatusOverride interface with the following assignment of parameters for each FreshnessValueId from the list in SOK_FM_1130:

[A: SOK_FM_1137]

valueId - the relevant FreshnessValueId (i.e. SecOc should be expected to be configured with SecOCOverrideStatusWithDataId == FALSE).

[A: SOK_FM_1138]

overrideStatus - SECOC_OVERRIDE_SKIP_UNTIL_NOTICE (0x43)

[A: SOK_FM_1139]

numberOfMessagesToOverride - 0xFF

[A: SOK_FM_1140]

When using this interface, it should be assumed that the SecOc module implements at least this interface according to AUTOSAR Release 19-11.

[A: SOK_FM_1141]

Subsequently, for each FreshnessValueId of an entry from SOK_fresh- ness_config_array to which a SecOCTxPduProcessing container belongs, the SOK-FM must call the SecOC_SendDefaultAuthenticationInformation interface with the following assignment of parameters:

[A: SOK_FM_1142]

FreshnessValueId - the relevant FreshnessValueId according to

[A: SOK_FM_1143]

SOK_FM_1141 sendDefaultAuthenticationInformation - TRUE

[A: SOK_FM_1149]

If SecOc has returned E_OK for at least one of the calls of SecOc_VerifyStatusOverride according to SOK_FM_1133 or SecOC_SendDefaultAuthenticationInformation according to SOK_FM_1141, SOK-FM must note in an internal variable that the error memory entry SOK_function_deactivated must be qualified (DEM_EVENT_STATUS_FAILED).

[A: SOK_FM_1203]

The evaluation of the internal variables from SOK_FM_1149 and if necessary the call of the DEM must take place according to SOK_FM_1202 in the next call of the SokFm_MainFunction().

[A: SOK_FM_1144]

If SecOC has returned E_NOT_OK on at least one of the calls of SecOc_VerifyStatusOverride according to SOK_FM_1133 and has returned E_NOT_OK on at least one of the calls of SecOC_SendDefaultAuthenticationInformation according to SOK_FM_1141, the function VW_SOK_ERROR_MULTIPLE_FAILURES returns.

[A: SOK_FM_1145]

Otherwise, if SecOC has returned E_NOT_OK on at least one of the calls to SecOc_VerifyStatusOverride according to SOK_FM_1133 and has returned E_OK on each call to SecOC_SendDefaultAuthen- ticationInformation according to SOK_FM_1141, the function returns VW_SOK_ERROR_DEACTIVATING_VERIFICATION_FAILED.

[A: SOK_FM_1146]

Otherwise, if SecOC has returned E_OK on each call to SecOc_VerifyStatusOverride according to SOK_FM_1133 and has returned E_NOT_OK on at least one of the calls to SecOC_SendDefaultAuthentica- tionInformation according to SOK_FM_1141, the function VW_SOK_ERROR_ACTIVATING_DEFAULT_AUTHENTICATION_FAILED returns.

[A: SOK_FM_1147]

A return of E_NOT_OK by SecOC on one of the calls SOK_FM_1133 or SOK_FM_1141 shall not cause the further calls not to be executed.

[A: SOK_FM_1148]

If SecOC has returned E_OK on every call to SecOc_VerifyStatusOverride according to SOK_FM_1133 and also returned E_OK on every call to SecOC_SendDefaultAuthenticationInformation according to SOK_FM_1141, the function returns E_OK.

[I: SOK_FM_1190]

The function of SOK-FM itself is not further restricted.

2.7.15 VW_SOK_FM_activate_SOK

[A: SOK_FM_1151]

- Input values: -
- Output values: -
- *Return values:* E_OK, 0x31 = VW_SOK_ERROR_ACTIVATING_VERIFICATION_FAILED, 0x33 = VW_SOK_ERROR_DEACTIVATING_DEFAULT_AUTHENTICATION_FAILED, 0x 34 = VW_SOK_ERROR_MULTIPLE_FAILURES.
- *Description:* This function is used to undo the deactivation from VW_SOK_FM_deactivate_SOK.

[A: SOK_FM_1152]

When this function is called, SOK FM must first compile the list of SecOcDataIds for which signature verification is to be switched on again. The same criteria apply for the creation of this list as for the list in SOK_FM_1130.

[A: SOK_FM_1153]

Subsequently, the SOK-FM must call the SecOc_VerifyStatusOverride interface with the following assignment of parameters for each SecOcDataId from the list in SOK_FM_1152:

[A: SOK_FM_1154]

valueId - the relevant SecOcDataId

[A: SOK_FM_1155]

overrideStatus - SECOC_OVERRIDE_CANCEL (0x02)

[A: SOK_FM_1156]

numberOfMessagesToOverride - 0xFF

[A: SOK_FM_1157]

Subsequently, for each FreshnessValueId of an entry from SOK_fresh- ness_config_array to which a SecOCTxPduProcessing-container belongs, the SOK-FM must call the SecOC_SendDefaultAuthenticationInformation interface with the following assignment of parameters:

[A: SOK_FM_1158]

FreshnessValueId - the relevant FreshnessValueId according to

[A: SOK_FM_1159]

SOK_FM_1157 sendDefaultAuthenticationInformation - FALSE

[A: SOK_FM_1160]

If SecOC has returned E_NOT_OK for at least one of the calls of SecOc_VerifyStatusOverride according to SOK_FM_1153 and returned E_NOT_OK for at least one of the calls of SecOC_SendDefaultAuthenticationInformation according to SOK_FM_1157, the function VW_SOK_ERROR_MULTIPLE_FAILURES returns.

[A: SOK_FM_1161]

Otherwise, if SecOC has returned E_NOT_OK on at least one of the calls to SecOc_VerifyStatusOverride according to SOK_FM_1153 and has returned E_OK on each call to SecOC_SendDefaultAuthen- ticationInformation according to SOK_FM_1157, the function returns VW_SOK_ERROR_ACTIVATING_VERIFICATION_FAILED.

[A: SOK_FM_1162]

Otherwise, if SecOC has returned E_OK on each call to SecOc_VerifyStatusOverride according to SOK_FM_1153 and has returned E_NOT_OK on at least one of the calls to SecOC_SendDefaultAuthentica- tionInformation according to SOK_FM_1157, then the function VW_SOK_ERROR_DEACTIVATING_DEFAULT_AUTHENTICATION_FAILED returns.

[A: SOK_FM_1163]

A return of E_NOT_OK by SecOC on one of the calls SOK_FM_1153 or SOK_FM_1157 shall not cause the further calls not to be executed.

[A: SOK_FM_1164]

If SecOC has returned E_OK for each call of SecOc_VerifyStatusOverride according to SOK_FM_1133 and has also returned E_OK for each call of SecOC_SendDefaultAuthenticationInformation according to SOK_FM_1141, then

[A: SOK_FM_1165]

- it must be noted in an internal variable of the SOK-FM that the error memory entry SOK_function_deactivated must be dequalified (DEM_EVENT_STATUS_PASSED).

[A: SOK_FM_1166]

- return the function E_OK.

[A: SOK_FM_1204]

The evaluation of the internal variables from SOK_FM_1165 and if necessary the call of the DEM must take place according to SOK_FM_1202 in the next call of the SokFm_MainFunction().

2.8 Integration against VKMS

[I: SOK_FM_1001]

SOK uses cryptographic keys provided by the VKMS system to generate and verify signatures.

[A: SOK_FM_1002]

The SOK-FM must operate the interfaces required by VKMS.

[A: SOK_FM_1018]

This dependency on VKMS must be able to be deactivated as an option at compile time, so that the integration of SOK-FM into a control unit does not depend on a previous integration of VKMS. In this case, all keys are always considered available.

[A: SOK_FM_1004]

The SOK-FM must call the VKMS_isTypeIdReady() function for each of these keys 100ms after the first call of the SokFm_MainFunction() function to determine whether the corresponding key has been contributed by VKMS and is available.

[A: SOK_FM_1006]

If at least one key is not available, the SOK-FM must qualify the error memory entry SOK_key_not_available (see SOK_FM_1008).

[A: SOK_FM_1005]

For each key that is not available, its type ID is stored in a list that can be read out via the SOK_missing_key_list measurement value (see SOK_FM_1013).

[A: SOK_FM_1011]

If the VKMS_isTypeIdReady() function has not returned any unique information about the availability of a key (e.g. because a key update is being performed in parallel), the SOK-FM must retry to obtain unique information from VKMS for this key at least every 100 milliseconds.

[A: SOK_FM_1003]

The SOK-FM must set up a callback function during its initialization to notify it of changes in the key material.

[A: SOK_FM_1010]

As soon as the callback function has been called, the SOK-FM must again perform a check of the keys configured for SOK for availability via a call of the function VKMS_isTypeIdReady() at the next call of the function SokFm_MainFunction() (see SOK_FM_1004 ff.).

[A: SOK_FM_1017]

The status of a key has no further influence on the behavior of the SOK-FM. In particular, it provides Freshness Value regardless of the status of the keys.

2.9 Measured values

[A: SOK_FM_965]

The SOK-FM uses the AUTOSAR DCM to output measured values.

[A: SOK_FM_966]

The exact definition of the measured values with their output formats is given in a separate table.

[I: SOK_FM_975]

The following names of the measured values correspond to the longnames of the SOK diagnostic table.

2.9.1 SOK_verification_failed_list

[A: SOK_FM_968]

This reading contains a list of all SecOC Data PDU ID whose verification state has been noted as failed at least once since initialization of the SOK-FM.

2.9.2 SOK_signature_failed_list

[A: SOK_FM_970]

This measurement value contains a list of all SecOC Data PDU ID for which signature generation by the SecOC module has failed at least once since initialization of the SOK- FM.

[A: SOK_FM_1026]

The SOK-FM detects that a signature generation has failed as soon as the timer of an associated Freshness Value started in SOK_FM_1020 exceeds a globally configurable threshold.

2.9.3 SOK_general_information

[A: SOK_FM_972]

This measured value contains information about the static configuration of the SOK-FM.

2.9.4 SOK_time_information

[A: SOK_FM_974]

This measured value contains runtime information about the SOK time.

[A: SOK_FM_1030]

If a control unit is configured neither as SOK time server nor as SOK time participant, the functions for providing the signals of the measured value SOK_time_information must nevertheless supply valid values. In particular, they must not supply a return value other than VW_SOK_OK for this reason only.

[A: SOK_FM_1031]

In this case, the signals of the measured value SOK_time_information must all be filled with zeros.

2.9.5 SOK_freshness_information

[A: SOK_FM_977]

This measurement value contains a list of runtime information for each configured Freshness Value.

2.9.6 SOK_missing_key_list

[A: SOK_FM_1013]

This measurement value contains a list of the type ID of the cryptographic keys that are to be used by the SecOC module within SOK and are reported as unavailable by VKMS (see SOK_FM_1001).

2.10 Diagnostic routines

2.10.1 SOK_function_deactivation

[A: SOK_FM_1136]

The specific interfaces for implementing this diagnostic routine must be designed in such a way that they conform to the interfaces expected by the DCM according to the AUTOSAR standard and the diagnostic data table provided by the customer.

[A: SOK_FM_1169]

If requirements are described below in such a way that certain parameters of the routine must be evaluated or filled, this is to be understood as meaning that the corresponding parameters of the interfaces defined according to SOK_FM_1136 must be evaluated or filled.

2.10.1.1 Routine_start

[A: SOK_FM_1167]

If the routine was started with parameter SOK_function == deactivate_sok_function (0x01), SOK-FM must call the internal function VW_SOK_FM_deactivate_SOK().

[A: SOK_FM_1170]

If the routine was started with parameter SOK_function == activate_sok_function (0x00), SOK-FM must call the internal function VW_SOK_FM_activate_SOK().

[A: SOK_FM_1174]

If the configuration value SOK_deactivation_development == TRUE, SOK-FM must persist the value passed in the parameter SOK_function in the NVM block provided for this purpose.

[A: SOK_FM_1176]

For this purpose, the persistence of this NvM block must also be explicitly triggered after the transfer to the NvM Manager.

2.10.1.2 Request_routine_results

[A: SOK_FM_1180]

If the configuration value SOK_deactivation_development == TRUE, SOK-FM must first check whether the persistence initiated in SOK_FM_1176 is complete and successful when retrieving the RoutineResult.

[A: SOK_FM_1181]

If persistence according to SOK_FM_1176 is not yet complete, the routine must be shown as still active.

[A: SOK_FM_1168]

If persistence according to SOK_FM_1176 was not required or is completed, the parameters of the routine result SOK_FM_1175, SOK_FM_1177, SOK_FM_1182, SOK_FM_1191 must be determined:

[A: SOK_FM_1175]

- If the function VW_SOK_FM_deactivate_SOK() or VW_SOK_FM_activate_SOK() returns a value other than E_OK, this value is to be output as SOK_deactivation_result.

[A: SOK_FM_1177]

- If E_OK was returned by function VW_SOK_FM_deactivate_SOK() or VW_SOK_FM_activate_SOK(), but the NvM manager returned an error when called after SOK_FM_1176 or the persistence of the NvM block failed, VW_SOK_ERROR_PERSISTING_DEACTIVATION_STATUS_FAILED is to be output as SOK_deactivation_result.

[A: SOK_FM_1182]

- If E_OK was returned by function VW_SOK_FM_deactivate_SOK() or VW_SOK_FM_activate_SOK() and persistence according to SOK_FM_1176 was successful or not required, E_OK is to be output as SOK_deactivation_result.

[A: SOK_FM_1191]

- If VW_SOK_FM_activate_SOK() was executed, 0x00 must be output as SOK_function_requested, otherwise if VW_SOK_FM_deactivate_SOK() was executed, 0x01 must be output as SOK_function_requested.

2.11 Event log entries

[A: SOK_FM_978]

The SOK-FM uses the AUTOSAR DEM to set event memory entries.

[A: SOK_FM_1029]

Debouncing of fault memory entries must be done by the DEM. Accordingly, the SOK-FM must pre-qualify an error memory entry at the DEM as pre-failed (DEM_EVENT_STATUS_PREFAILED) or pre-sporadize it as pre-passed (DEM_EVENT_STATUS_PREPASSED).

[A: SOK_FM_979]

The exact definition of the event memory entries is done in a separate table.

[A: SOK_FM_980]

The following names of the event memory entries correspond to the DTC-DFCC text of the SOK diagnostic table.

[A: SOK_FM_1201]

All SOK event memory entries may only be (pre)qua- lified and (pre)dequalified in the context of the SokFm_MainFunction.

[A: SOK_FM_1202]

If the conditions for (de)qualifying an event memory entry are checked in a context other than the SokFm_MainFunction(), the status must first be stored in an internal variable and the DEM interface must then be called in the next SokFm_MainFunction call.

2.11.1 SOK_timeserver_not_available

[A: SOK_FM_982]

If the function SokFm_TriggerCrRequestSecuredSOKTime() has been called successfully (see SOK_FM_807, VW_SOK_812) without an authentic SOK- time having been received from the SOK time server between the calls, the SOK-FM must pre-qualify the event memory entry SOK_timeserver_not_available.

[A: SOK_FM_983]

As soon as an authentic SOK time has been received from the SOK time server, the SOK-FM must pre-sporadize the event memory entry.

2.11.2 SOK_signature_verification_failed

[A: SOK_FM_985]

As soon as the verification of the signature of a received SOK message fails, the SOK FM shall mark the verification state of the corresponding SecOC Data PDU ID as failed.

[A: SOK_FM_986]

If the verification state of at least one SecOC Data PDU ID is noted as failed, the SOK-FM must pre-qualify the SOK_signature_verification_failed event store entry in the next call to the SokFm_MainFunction.

[I: SOK_FM_987]

The corresponding Data PDU ID can be read out via a separate measured value (see SOK_FM_968).

[A: SOK_FM_989]

As soon as the verification of the signature of a received SOK message is successful, the SOK-FM shall mark the verification state of the corresponding SecOC Data PDU ID as successful.

[A: SOK_FM_990]

If the verification state is not marked as failed for any SecOC Data PDU ID, the SOK-FM must pre-sporadize the SOK_signature_verification_failed event store entry in the next call to the Sokfm_MainFunction if it is currently qualified.

[A: SOK_FM_991]

If the event memory entry SOK_signature_verification_failed is deleted (e.g. manually or by a sufficient number of ECU restarts), the SOK-FM must clear the list of failed verifications of the measured value SOK_verification_failed_list.

2.11.3 SOK_key_not_available

[A: SOK_FM_1008]

If a call of the SOK-FM of the function VKMS_isTypeIdReady() has revealed that at least one key required for SOK is not available (see SOK_FM_1001), the SOK-FM must pre-qualify the event memory entry SOK_key_not_available.

[A: SOK_FM_1009]

As soon as the SOK-FM determines that all required keys are available, it must pre-sporadize the erroneous entry.

2.11.4 SOK_function_deactivated

[I: SOK_FM_1172]

This event memory fragment indicates that the functionality of SOK has been (partially) deliberately deactivated.

[I: SOK_FM_1173]

The check, whether the conditions for (de)qualification of the event memory entry are given, is done within the functions VW_SOK_FM_deactivate_SOK() or VW_SOK_FM_acti- vate_SOK().

3 Appendix

3.1 Change documentation

[A: SOK_FM_45]

The change documentation listed here only gives a rough overview of the changes made between versions.

[I: SOK_FM_3.]

Table 1: Change documentation

Datum	Version	Kapitel	Beschreibung	Autor/OE
19.12.2016	0.10	-	Erstellung	Tschache, Alexander (EEKS)
20.03.2017	0.11	-	Revision nach erster Software-Auslieferung Messwerte, Ereignisspeichereinträge	Tschache, Alexander (EEKS/5)
27.03.2017	1.0	-	Integration gegen VKMS; die entsprechenden Messwerte; Naxis-spezifikation des neuen CR-Paradigmas	Tschache, Alexander (EEKS/5)
27.03.2017	1.1	-	Bigendian anstelle von little endian; Startup-Verhalten bezüglich SOK-Zeit	Tschache, Alexander (EEKS/5)
19.04.2017	1.2	2.7.4	Vorzeichenfehler bei Jitter-Berechnung	Tschache, Alexander (EEKS/5)
13.06.2017	1.3	2.10.3 2.6	Anpassung für Fehlerbehandlung bei fehlenden Schlüsseln	Tschache, Alexander (EEKS/5)
11.01.2018	1.4	2.2.1.1 2.7.8 2.10	Konfigurationsoption für Deaktivierung des Replayschutzes; Anpassung Fehlerqualifizierung	Tschache, Alexander (EEKS/5)
25.02.2019	1.5	2.9, 2.10	Korrektur Fehlerentprellung und Messwerte für SOK-Zeit	Tschache, Alexander (EEKS)
16.05.2019	2.0	-	Einführung Session-basierter Freshness Values	Tschache, Alexander (EEKS/1)
18.06.2021	2.A	div.	Entroduction Ability via Diagnosis routine	Fernandez, Aljoscha (EESC/3)
18.10.2021	2.A.1	div.	Detailkorrekturen (Namen Diagnoseobjekte) Ability	Fernandez, Aljoscha (EESC/3)
26.04.2022	2.2	div.	Verification StatusOverride About Freshness Id, Anpassung Vget Default Zeit, remove DTC signature_creation, DTCs only from Main Function	Gierds, Christian (TS-22)

3.2 Referenced documents

[I: SOK_FM_5]

Table 2: Referenced documents

Ref.	Document/ Source	Version
/1/	LAH 893 .909 .D - Group Groundsanforderungun gen Softw are	2 .3
/2/	VW 80180 -1 - Cyptographic algorithms and methods for use in special devices, systems, functions, and supplier-specific volumes	