Document Title	Specification of Intrusion Detection System Manager
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	977

Document Status	published
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	R20-11

Document Change History			
Date	**Release**	**Changed by**	**Description**
2020-11-30	R20-11	AUTOSAR Release Management	• Initial release

Disclaimer

Table of Contents

1 Introduction and functional overview

This specification describes the functionality, API, and the configuration for the AUTOSAR Basic Software module **Intrusion Detection System Manager** (**IdsM**).

The IdsM is part of the AUTOSAR Intrusion Detection System (**IDS**).

An overview and description of the elements of a distributed IDS according to AUTOSAR is available in the IDS requirement specification [1].

The software component IdsM provides a standardized interface for receiving notifications of on-board security events SEv. The SEvs can be reported by security sensors implemented in Basic Software Modules (BSW) and application Software Components (SW-C).

Additionally, the SEvs can be reported with optional context data such as event type and suspicious data, which can be useful information for the security forensic performed at the backend.

Besides collecting, the IdsM has the capability of qualifying SEvs according to configurable rules. The IdsM filters and transforms reported SEvs to qualified on-board security events (**QSEv**). The QSEv are further handled by the IdsM for storage or forwarding.

Depending on the overall security concept, QSEv can be persisted locally on the ECU via Security Event Memory (**Sem**), propagated towards configured sinks, or both. The available sinks are the Diagnostic Event Manager (**Dem**) module and the IDS Reporter Module (**IdsR**), which might pass the QSEv data to a security operation center (**SOC**) in the backend.

2 Acronyms and Abbreviations

The glossary below includes acronyms and abbreviations relevant to the Intrusion Detection System Manager module that are not included in the AUTOSAR_TR_Glossary [2].

Abbreviation / Acronym:	Description:
API	Application Programming Interface
BSW	Basic Software
BswM	Basic Software Mode Manager
CDD	Complex Device Driver
Classic Platform	AUTOSAR Classic Platform
Csm	Crypto Service Manager
Dcm	Diagnostic Communication Manager
Dem	Diagnostic Event Manager module
DET	Default Error Tracer
ECU	Electronic Control Unit
ECUC	ECU configuration
ID	Identifier
IDS	Intrusion Detection System
IdsM	Intrusion Detection System Manager
IdsR	Intrusion Detection System Reporter
IF	Interface
MCU	Microcontroller Unit
NvM	Non-volatile memory
NVRAM	Non-volatile random access memory
OEM	Original Equipment Manufacturer
PDU	Protocol Data Unit
PDU ID	PDU Identifier
PduR	PDU Router
QSEv	Qualified Security Event
RTE	Runtime Environment
SecXT	Security Extract
Sem	Security Event Memory
SEv	On-board Security Event
StbM	Synchronized Time-Base Manager
SW-C	Software Component
SOC	Security Operation Center
TP	Transport Protocol

Terms:	Description:
Context Data Buffer	Buffer with variable sizes to fit to the needs of the context data of the SEvs.
Context Data	Relevant information to a SEv. It is optional data that provides a broader understanding of the security event (e.g. the corrupted data). The content and encoding of the context data is externally defined by the sensor and unknown to the IdsM module.
Event Buffer	Buffer to temporarily store the reported SEv IDS.
Filter	A modifier of the security events which can drop or alter an incoming SEv.
Filter Chain	One configured sequence of filters.

Terms:	Description:
IdsM block state	State reported by the BswM via IdsM_BswM_StateChanged. The states are used to suspend the collection of security events.
Intrusion Detection System Manager	The Intrusion Detection System Manager handles security events reported by security sensors.
IdsR	The IdsR is an OEM specific adaptive application that can be used to further propagate the QSEvs to the SOC.
Qualified Security Event (QSEv)	Events that have passed their corresponding filter chain and are sent to the configured sink.
Security Event (SEv)	On-board Security Events are instances of security event types which are reported by BSW or SW-C to the IdsM. They are structured data originating from a sensor which serve as fundamental input and output data format for filters. These reported events to the IdsM that are indicative of an ongoing attack or are somehow suited to assess the security state of the vehicle. This means that events can occur during the normal operation without any ongoing attack.
Security Event Type	A security event type can be identified by its security event type ID. Instances of security event types are called security events and share the same security event type ID.
Sem	Security event memory is a Dem Module user defined memory which is separated from the Dem's primary memory.
Sensor	Reporting identity that informs the IdsM module about SEvs. It can be a BSW Module, a proprietary CDD or an SW-C Application.
Sink	Destination of a QSEv. Depending on the configuration the QSEv can be persisted, propagated or both.
Timestamp Provider	Service or SW-Component which provides a TimeStamp. e.g. in CP Stbm.

3 Related documentation

3.1 Related Specification

AUTOSAR provides a General Specification on Basic Software modules [3], which is also valid for the Intrusion Detection Manager.

Thus, the specification SWS BSW General shall be considered as additional and required specification for the Intrusion Detection Manager.

This document is part of the AUTOSAR IDS specification and covers aspects specific to `Classic Platform` only. For other aspects of the IDS specification, please refer to the following documents:

- **AUTOSAR_RS_Intrusion Detection System [1]:** Specifies IDS system requirements.
- **AUTOSAR_PRS_IntrusionDetectionSystem [4]:** Specifies the communication protocol for the transmission of security events.
- **AUTOSAR_MOD_GeneralDefinitions [5]:** Standardized Security Events reported by AUTOSAR BSW
- **AUTOSAR_TPS_SecurityExtractTemplate [6]:** Specifies the Security Extract.

3.2 Input Documents & Related Standards and Norms

[1] Requirements on Intrusion Detection System
AUTOSAR_RS_IntrusionDetectionSystem

[2] Glossary
AUTOSAR_TR_Glossary

[3] General Specification of Basic Software Modules
AUTOSAR_SWS_BSWGeneral

[4] Specification of Intrusion Detection System Protocol
AUTOSAR_PRS_IntrusionDetectionSystem

[5] Standardized M1 Models used for the Definition of AUTOSAR
AUTOSAR_MOD_GeneralDefinitions

[6] Security Extract Template
AUTOSAR_TPS_SecurityExtractTemplate

[7] General Requirements on Basic Software Modules
AUTOSAR_SRS_BSWGeneral

4 Constraints and assumptions

[SWS_IdsM_CONSTR_00001] ⌈The Intrusion Detection Manager has no knowledge of the meaning of the *Context Data* reported within a SEv; thus, it can not determine independently if a system has being compromised or not. Identification and threat response is realized outside of the scope of IdsM, e.g., in a SOC.⌋*()*

4.1 Assumptions

The following assumptions have been made in the design of the IdsM concept:

- **Precision of timestamps:** The timestamps of events received by the backend may be inaccurate to some degree. However, it shall be possible in most cases to extract the order of events from the events received by the backend. In some cases, this might not be possible, e.g., because of events occurring in parallel on different ECUs or because of inherent tolerances in time synchronization.
- **Uniqueness of QSEv:** Events do not need to be uniquely identifiable. Two events may contain the same data.
- **Dropping of events:** It is acceptable that SEvs are dropped depending on their reporting frequency and criticality, e.g., a general overload of the system.
- **Semantics of events:** Security-related events are indicative of a potential ongoing attack or are somehow suited to assess the security state of the vehicle. Meaning that events can occur during the normal operation without any attack happening.

4.2 Applicability to Car Domains

The AUTOSAR Intrusion Detection System Manager is generic and provides flexible configuration. It is independent of the underlying communication system and can be applied to any automotive domain under limitations and assumptions provided above.

5 Dependencies to other modules

5.1 Interfaces to Modules

The AUTOSAR **Intrusion Detection System Manager** includes header files of the modules BswM, Dcm, DET, Dem, NvM, PduR, and the RTE. Furthermore, it provides generic interfaces to Basic Software Modules and Software Components *(Sensors)* for reporting their SEvs.

Figure 5.1 shows the interfaces provided to and required from other modules in the AUTOSAR BSW.

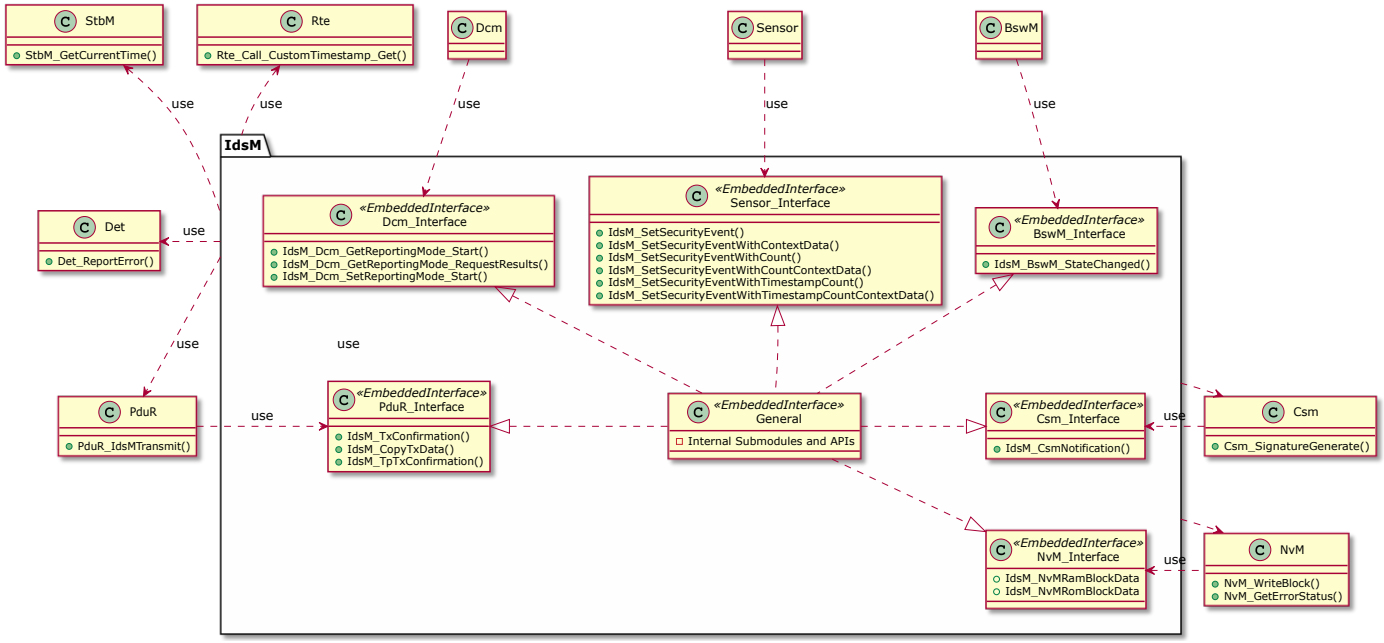

Figure 5.1: IdsM's interfaces to other modules

5.1.1 Sensor Modules

The IdsM provides generic IdsM interfaces that notify Security Events (SEvs) with additional information depending on the configuration.

Standard API Used by the Basic Software Modules and by Software Components.

- Notification of a SEv
- Notification of a SEv with context data

Smart Sensor API Used by software components in cases in which it is necessary to transmit an event count and a timestamp. These additional parameters are already calculated by a smart sensor. They are located either in a SW-C or a Cdd.

- Notification of a SEv with a counter

- Notification of a SEv with a counter and context data
- Notification of a SEv with a timestamp and a counter
- Notification of a SEv with a timestamp, a counter and context data

5.1.2 Error Handling Modules

IdsM reports development errors to the Default Error Tracer.

5.1.3 Diagnostic Access

The Dcm module is able to modify the configuration of the events' reporting level.

5.1.4 Persistence of Reporting Level

The NvM module persists the configuration values of the events' reporting level.

5.1.5 IdsR Sink

The PduR is used in case the events are configured to be sent to the IdsR sink; The sending of the events is bus independent.

5.1.6 Dem / Sem Sink

The Dem module is used in case the events are configured to be logged in the Dem / Sem sink.

5.1.7 BSW Scheduler

The IdsM needs cyclic invocation of its main scheduling function in order to evaluate and handle the reported SEvs.

5.2 File Structure

This section explains the file structure of the IdsM.

5.2.1 Code File Structure

For details, refer to the section 5.1.6 “Code file structure” in [3, SWS BSW General].

5.2.2 Header File Structure

Besides the files defined in section 5.1.7 “Header file structure” in [3, SWS BSW General], the Intrusion Detection System Manager module needs to include the files defined below.

[SWS_IdsM_00101] ⌈The IdsM module shall include the header file Det.h if the parameter IdsMDevErrorDetect is enabled.⌋*()*

[SWS_IdsM_00102] ⌈The IdsM module shall include the header file Dem.h if the parameter IdsMSinkDem is enabled.⌋*()*

[SWS_IdsM_00103] ⌈The IdsM module shall include the header file Dcm.h if the parameter IdsMDiagnosticSupport is enabled.⌋*()*

[SWS_IdsM_00104] ⌈The IdsM module shall include the header file NvM.h if the parameter IdsMNvmBlockDescriptor is configured.⌋*()*

[SWS_IdsM_00105] ⌈The IdsM module shall include the header file PduR.h if the parameter IdsMSinkIdsR is enabled.⌋*()*

6 Requirements Tracing

The following tables reference the requirements specified in the IDS requirement specification [1], the Specification of Intrusion Detection System Protocol [4, Specification of Intrusion Detection System Protocol] and BSW General system requirement specification [7] 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
[DRAFT]	No description	[SWS_IdsM_01600] [SWS_IdsM_01601] [SWS_IdsM_01602]
[RS_IDS_00300]	Provide configurable filter chains for qualifying SEv	[SWS_IdsM_01001] [SWS_IdsM_01003] [SWS_IdsM_01004] [SWS_IdsM_01005]
[RS_IDS_00301]	Provide multiple filter chains	[SWS_IdsM_01001]
[RS_IDS_00310]	Configure reporting mode per Security Event Type and IdsM instance	[SWS_IdsM_01012] [SWS_IdsM_01013]
[RS_IDS_00320]	Support machine state filter	[SWS_IdsM_01023]
[RS_IDS_00330]	Support sampling filter	[SWS_IdsM_01031] [SWS_IdsM_01032]
[RS_IDS_00340]	Support Aggregation filter	[SWS_IdsM_01041] [SWS_IdsM_01043] [SWS_IdsM_01044] [SWS_IdsM_01045] [SWS_IdsM_01046] [SWS_IdsM_01047] [SWS_IdsM_01048] [SWS_IdsM_01049]
[RS_IDS_00350]	Support Threshold filter	[SWS_IdsM_01061] [SWS_IdsM_01062]
[RS_IDS_00502]	Event Timestamps	[SWS_IdsM_01106]
[RS_IDS_00503]	Timestamp Sources	[SWS_IdsM_01107] [SWS_IdsM_01108] [SWS_IdsM_01109] [SWS_IdsM_01110] [SWS_IdsM_01112]
[RS_IDS_00505]	Authenticity of QSEvs	[SWS_IdsM_01204]
[RS_IDS_00510]	The IdsM shall allow to transmit QSEv to the IdsR	[SWS_IdsM_01203]
[RS_IDS_00511]	Limit event rate and traffic	[SWS_IdsM_01070] [SWS_IdsM_01081] [SWS_IdsM_01091]
[RS_IDS_00610]	Configuration of qualification filters for SEv	[SWS_IdsM_01002]

7 Functional specification

7.1 Overview

The Intrusion Detection functionality consists of collecting possible security events, handle them with filter rules and forward them towards configured sinks.

This chapter specifies the functional behavior of the IdsM for the Classic Platform.

Figure 7.1 shows how the IdsM is integrated in the AUTOSAR BSW security stack:

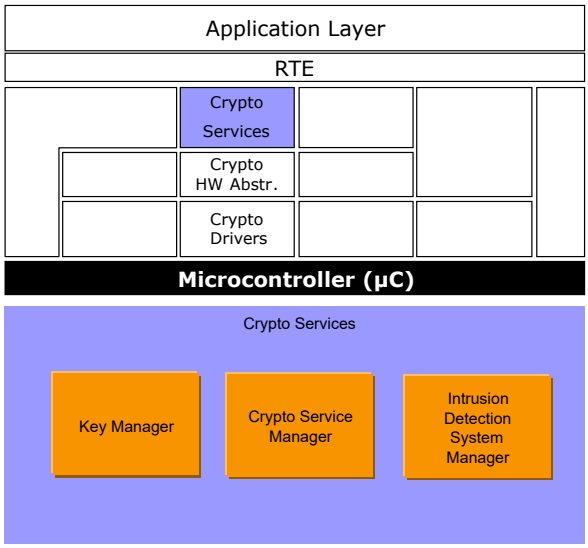

Figure 7.1: AUTOSAR BSW architecture showing the IdsM module

The modules that act as sensors and report SEvs towards the IdsM are:

- AUTOSAR Basic Software Modules (BSW)
- Proprietary Complex Device Drivers (CDD)
- Application Software Components (SW-C)

The collected On-board Security Event SEvs are processed by a series of configured rules called "Filter Chains" into QSEvs, which can be sent to the following sinks:

- Intrusion Detection System Reporter (IdsR), using the PduR for transmission of the QSEvs.
- Dem / Sem Module, for local persistence of the QSEv records.

It is possible to reconfigure specific event parameters and filter qualifiers via diagnostics using the Dcm module. 7.10.1

Optionally integrity and confidentiality of the QSEv records can be enforced via crypto-algorithms.

Figure 7.2 shows the interaction with the modules mentioned above. The modules CanIf, LinIf, EthIf, KeyM and SecOC are illustrated as BSW sensor examples.

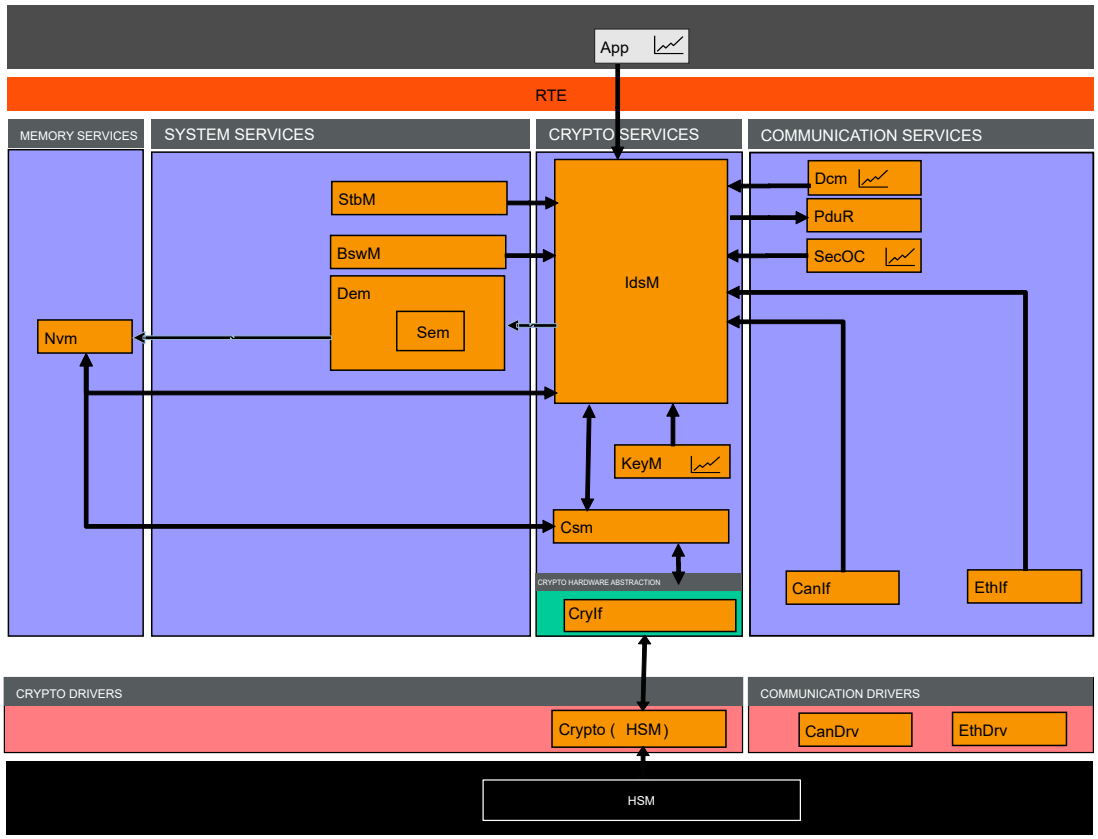

Figure 7.2: Interaction of the IdsM with other stack modules

7.2 Module Handling

The functionality of the `IdsM` is divided into the following functional sub-modules:

- Reception of Events
- Buffering of Events
- `IdsM` Internal SEvs
- Qualification of Events
- Reporting of QSEvs
- Persistence of specific parameter of events in NvM
- Read and Write specific parameters of events via diagnostics with Dcm

Figure 7.3 shows the allocation in the stack of the functional sub-modules listed above, these are described in detail throughout this chapter 7.

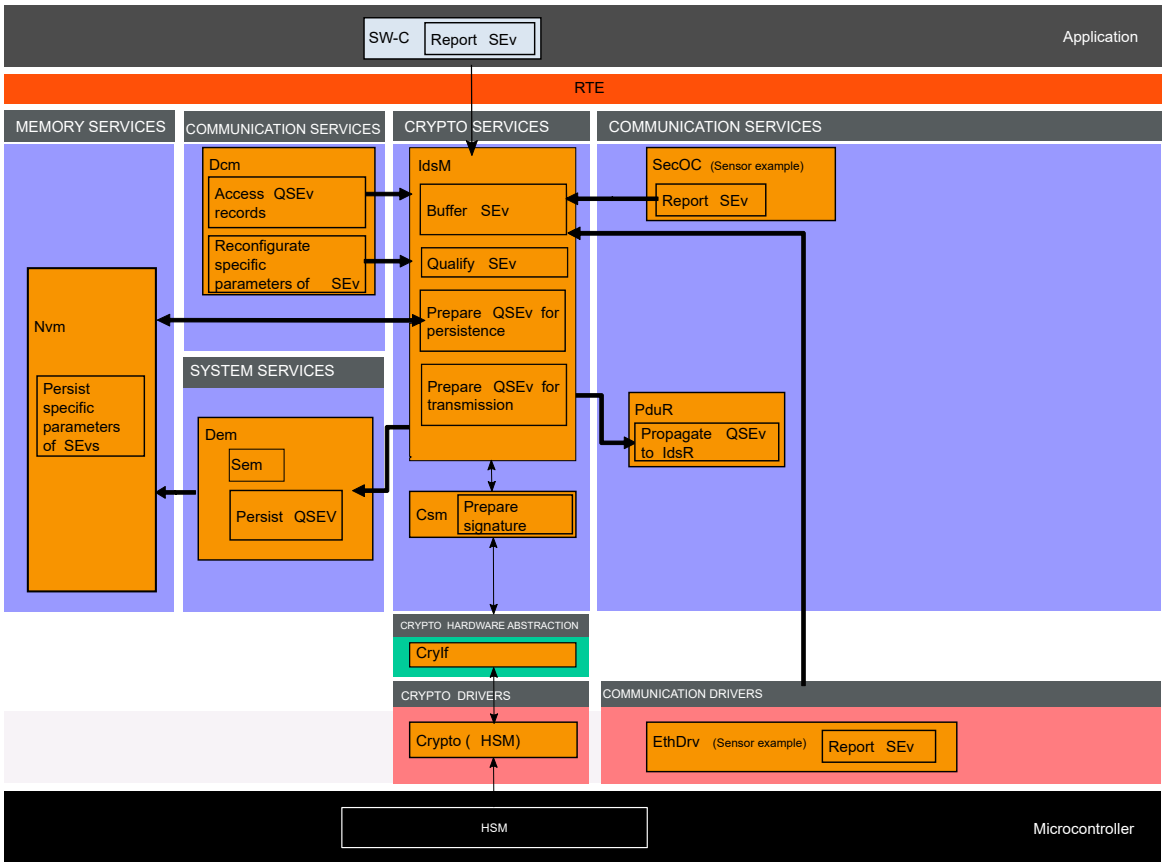

Figure 7.3: Functional modules of the IdsM.

7.2.1 Initialization

The IdsM module is initialized via IdsM_Init. Except for IdsM_GetVersionInfo and IdsM_Init, the API functions of the IdsM module may only be called after the module has been properly initialized.

[SWS_IdsM_00202] ⌈A call to IdsM_Init initializes all internal variables and sets the IdsM module to the initialized state.⌋*()*

[SWS_IdsM_00203] ⌈If development error reporting is enabled via IdsMDevErrorDetect, the IdsM module shall call Det_ReportError with the error code IDSM_E_PARAM_UNINIT when any API other than IdsM_Init or IdsM_GetVersionInfo is called in uninitialized state.⌋*()*

[SWS_IdsM_00204] ⌈When IdsM_Init is called in initialized state, the IdsM module shall not re-initialize its internal variables. It shall instead call Det_ReportError with the error code IDSM_E_ALREADY_INITIALIZED if development error reporting is enabled (see IdsMDevErrorDetect).⌋*()*

7.2.2 Timing Related Functionality

To be able to handle the security events and their filters asynchronously, the IdsM module is triggered cyclically via the IdsM_MainFunction.

7.3 Reception and Buffering of Events

7.3.1 Reception of Events

If a sensor reports a security event via the IdsM services IdsM_SetSecurityEvent or IdsM_SetSecurityEventWithContextData, without and with context data respectively, an event buffer from the IdsM event buffer pool is used and processed asynchronously in the IdsM_MainFunction function.

If context data exists, a context data buffer with the adequate size will be used. If there are currently no context buffers available, the event is processed without context data.

The service IdsM_SetSecurityEvent and IdsM_SetSecurityEventWithContextData can be used by any sensor, independently of its source.

[SWS_IdsM_00300] ⌈The IdsM shall be able to receive SEvs with the service IdsM_SetSecurityEvent when there no context data is reported.⌋*()*

[SWS_IdsM_00301] ⌈The IdsM shall be able to receive SEvs with the service IdsM_SetSecurityEventWithContextData when the optional context data is reported.⌋*()*

7.3.1.1 Smart Sensors

Smart sensors provide additional information to the standard sensors. The smart sensors can be a SW-C or a CDD which previously records a timestamp and calculates a counter for a certain SEv Type. The services in this section are available though:

- Service interfaces for the SW-Cs. (Refer to: 7.8.2.2).
- Direct C API call for the CDDs.

Reception of SEvs with Counter

[SWS_IdsM_00401] ⌈The IdsM shall be able to receive SEvs with a counter calculated from a smart sensor with the service IdsM_SetSecurityEventWithCount⌋*()*

[SWS_IdsM_00402] ⌈The IdsM shall be able to receive SEvs with a counter calculated from a smart sensor, and additionally the SEv context data, with the service IdsM_SetSecurityEventWithCountContextData⌋*()*

Reception of SEvs with Counter and Timestamp

[SWS_IdsM_00403] ⌈The IdsM shall be able to receive SEvs with a timestamp and a counter calculated from a smart sensor with the service IdsM_SetSecurityEventWithTimestampCount⌋*()*

[SWS_IdsM_00404] ⌈The IdsM shall be able to receive SEvs with a timestamp, a counter calculated from a smart sensor,and additionally the SEv context data, with the service IdsM_SetSecurityEventWithTimestampCountContextData⌋*()*

[SWS_IdsM_00405] ⌈For reporting a SEvs with a timestamp but with no previously calculated counter, the services IdsM_SetSecurityEventWithTimestampCountContextData and IdsM_SetSecurityEventWithTimestampCount can be used with the counter value equals 1.⌋*()*

Context Data Details

[SWS_IdsM_00501] ⌈The functions IdsM_SetSecurityEventWithContextData, IdsM_SetSecurityEventWithCountContextData and IdsM_SetSecurityEventWithTimestampCountContextData shall support a maximum length of 1500 bytes for the context data.⌋*()*

Note: To avoid overloading of the network, a maximum of 1500 bytes for the context data is recommended, especially when transmitting on CAN Bus.

There might be cases in which this limit is insufficient to transmit all the sensor's information, in that case it shall be evaluated that there are enough resources to avoid flooding of the communication channels.

[SWS_IdsM_00502] ⌈The functions calling IdsM_SetSecurityEventWithContextData, IdsM_SetSecurityEventWithCountContextData and IdsM_SetSecurityEventWithTimestampCountContextData shall provide the context data pointer to a byte array where the context data is available.⌋*()* **Note:** The IdsM shall not rely on knowledge of the internal structure of the optional context data.

7.3.2 Security Event Definition

A Security Event or Security Event Instance SEv defines the atomic unit, reported by a sensor, that can be handled by the IdsM module. The IdsM receives the notification of a sensor from BSW or CDD modules, or from SW-Cs via the RTE. The IdsM module uses the EventId to manage the status of the SEv of a system and performs the required actions for individual results, e.g., filtering, storing, reporting via the network. A Security Event Definition represents the type of event to be reported. The definition, found in the SecXT, includes a global unique identifier and the short-name of the reporting module.

[SWS_IdsM_00600] ⌈The IdsM module shall represent each SEv instance by an IdsMExternalEventId, a IdsMSensorInstanceId, a IdsMInternalEventId, and the related EventName. These combination of parameters shall be unique per IdsM instance represented by the ECU configuration.⌋*()*

[SWS_IdsM_00601] ⌈Each SEv shall have an IdsMInternalEventId. This parameter shall not be configured manually. The IdsM shall calculate the value of this parameter internally and shall publish the value in the parameter. This ID is used for internal handling of the SEvs.⌋*()*

[SWS_IdsM_00602] ⌈Sensors using the IdsM API to report SEvs shall not rely on the value of the parameter IdsMInternalEventId. Instead, they shall use the symbolic constant (SymbolicNameValue) of the corresponding SEv.⌋*()*

[SWS_IdsM_00603] ⌈Each SEv shall have an external event ID IdsMExternalEventId, which is a global and unique ID per Security Event Type represented by the ECU configuration, and it is defined in the SecXT.⌋*()*

[SWS_IdsM_00604] ⌈A IdsMExternalEventId with value 0xFFFF shall be considered invalid.⌋*()*

All sensors use the symbolic name of their corresponding IdsMEvent Container as identifier to report their SEvs. When generating the dynamic code, the symbolic names are replaced by numbers (the calculated number is published as internal event ID). The generated symbolic name represents the tuple of an external event id and a sensor instance id. This ID is used for internal handling of the SEvs.

[SWS_IdsM_00605] ⌈Each SEv shall have a sensor instance ID IdsMSensorInstanceId. This is the representation of the module number, in case there are many instances of the same module reporting to the IdsM.⌋*()*

[SWS_IdsM_00606] ⌈The combination of external event ID IdsMExternalEventId and sensor instance ID IdsMSensorInstanceId shall make the SEvs uniquely identifiable within the configuration. This parameter tuple is represented by the *Symbolic Name Value* of the IdsMEvent Container.⌋*()*

[SWS_IdsM_00607] ⌈Sensors using the IdsM services shall report a SEv using the symbolic constant (SymbolicNameValue) of the IdsMEvent Container .⌋*()*

The IdsM is designed to handle the case where more than one SEv shares the same IdsMExternalEventId as long as the reporting modules have unique sensor instance Id.

[SWS_IdsM_00608] ⌈Each SEv shall have a IdsMSensorInstanceId configured. In case there are several instances of the same sensor reporting SEvs with the same *Event Definition ID* in a ECU, the reporting entity shall be uniquely identified through the configuration parameter IdsMSensorInstanceId. In case there is only one instance of the module in the configuration, the value of the instance ID shall be, by default, set to 0.⌋*()*

7.3.3 Buffers

[SWS_IdsM_00701] ⌈The IdsM shall have a configurable number of event buffers IdsMNumberOfEventBuffers, depending on the amount of configured IdsMEvents that are to be handled.⌋*()*

A recommended number of buffers can be calculated as follows:
Number of Event Buffers = Number of Event Aggregation Filter instances + Upper bound of parallel processed events

[SWS_IdsM_00702] ⌈Upon reception of a SEv, the IdsM shall store the event in an Event Buffer until it can be further processed. Event buffers shall be handled and filtered asynchronously in the IdsM_MainFunction service.⌋*()*

[SWS_IdsM_00703] ⌈In case no Event Buffer is found. The IdsM internal SEv **'No Event Buffer Available'** shall be triggered, in case it has been configured.⌋*()* See IDSM_INTERNAL_EVENT_NO_EVENT_BUFFER_AVAILABLE in [SWS_IdsM_91015].

[SWS_IdsM_00704] ⌈The IdsM shall have a configurable number of context data buffers IdsMNumberOfContextDataBuffers with different configurable sizes IdsMContextDataBufferSize in order to satisfy different sensor use cases.⌋*()*

Rationale: There can be significant differences in the size of the context data depending on the type of event being processed. These sizes have to be configured suitably to utilize the memory resources effectively.

[SWS_IdsM_00705] ⌈Upon reception of a SEv with context data, the IdsM shall store the context data in an Context Data Buffer with the most adequate size available. A configured *Context Data Buffer Pool* shall be searched in order to find a buffer with the same size as the reported context data, or find the next larger buffer. These buffers shall be handled and filtered asynchronously in the IdsM_MainFunction service.⌋*()*

[SWS_IdsM_00706] ⌈Once an appropriate Context Data Buffer has been found, it shall be linked to the corresponding Event Buffer for further processing.⌋*()*

[SWS_IdsM_00707] ⌈In case there is no appropriate Context Data Buffer of the same size or larger than the context data, the event shall be processed as an event without context data. Thus no context data buffer shall be linked to the processed SEv.⌋*()*

[SWS_IdsM_00708] ⌈In case no appropriate Context Data Buffer is found. The IdsM internal SEv **'No Context Data Buffer Available'** shall be triggered, in case it has been configured.⌋*()* See IDSM_INTERNAL_EVENT_NO_CONTEXT_DATA_BUFFER_AVAILABLE in [SWS_IdsM_91015].

[SWS_IdsM_00709] ⌈Upon reception of a SEv with no context data, the IdsM shall not use any Context Data Buffer. Thus no context data buffer will be linked to the processed SEv.⌋*()*

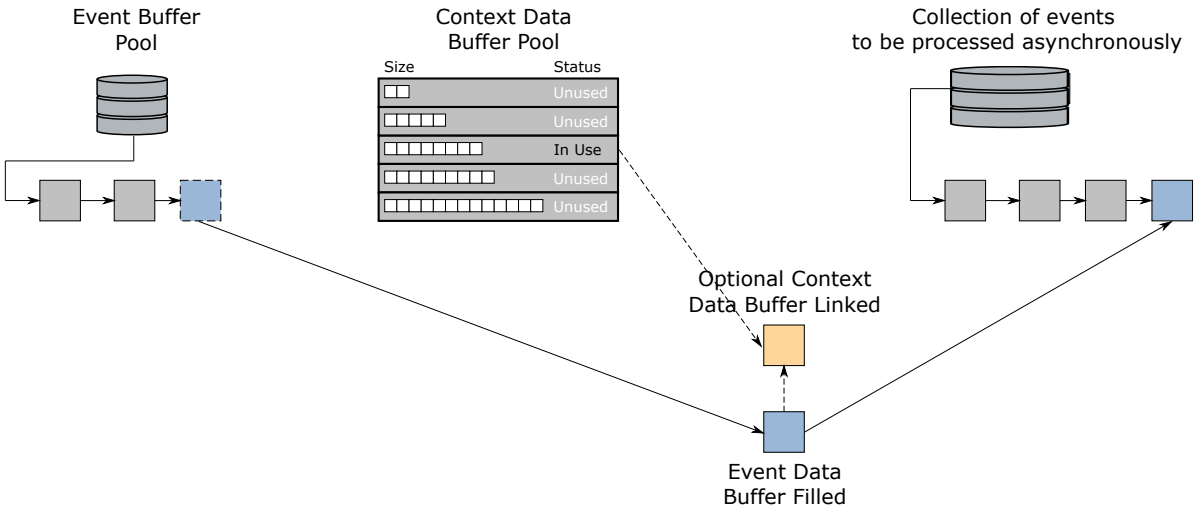

Figure 7.4: Use of Event Buffers and Context Data Buffers

Figure 7.4 shows how event buffers and context buffers are used when security events are set via the IdsM API as this chapter explains. Upon reception of a SEv, the event is stored in a buffer from the event buffer pool and its corresponding context data is stored in a buffered of the most adequate size from the context data buffer pool. These two buffers are linked and processed together asynchronously in the IdsM_MainFunction service.

7.4 IdsM Internal SEvs

The module IdsM itself can also be used as a Security Event sensor.

[SWS_IdsM_00801] ⌈The security events reported by IdsM module are listed in [SWS_IdsM_91015].⌋*()*

[SWS_IdsM_91015] Security events for IDSM ⌈

Name	*Description*	*ID*
IDSM_INTERNAL_EVENT_NO_EVENT_BUFFER_AVAILABLE	A SEv cannot be handled because there are no more event buffers available to process the event.	46
IDSM_INTERNAL_EVENT_NO_CONTEXT_DATA_BUFFER_AVAILABLE	The context data of an incoming event cannot be stored because there are no more context data buffers available.	47
IDSM_INTERNAL_EVENT_TRAFFIC_LIMITATION_EXCEEDED	The current traffic exceeds a configured traffic limitation.	48

⌋*()*

[SWS_IdsM_00802] ⌈In case the IdsM Internal Events are configured, the IdsM shall provide own buffers for each one of these SEvs. These are dedicated buffers, independent from the common Event Buffers used for normal SEvs.⌋*()*

Note: Having dedicated buffers allows the IdsM to inform the sink about malfunctioning even if the IdsM is overloaded.

[SWS_IdsM_00803] ⌈IdsM internal SEvs shall not be filtered by IdsM instance specific filters.⌋*()* See 7.5 for filter categories.

[SWS_IdsM_00804] ⌈IdsM internal SEvs can be filtered by IdsM SEvID specific filters.⌋*()* See 7.5 for filter categories.

7.5 Qualification of SEvs

Raw Security Events can be generated at a very high rate by the BSW. However, only a subset of these events might be of interest to the OEM. By preprocessing the raw SEv and dropping all events that do not match the filtering criteria, resource needs on the ECUs and the network can be reduced.

[SWS_IdsM_00901] ⌈The IdsM shall store the SEvs in the Event Buffers and process them asynchronously in the IdsM_MainFunction service in order to identify them as QSEvs.⌋*()*

[SWS_IdsM_00902] ⌈The qualification of reported security events shall take place by evaluating the processed SEv against a configurable sequence of filters, known as the filter chain.⌋*()*

[SWS_IdsM_00903] ⌈A SEv shall contain the information of the filter chain that is used to qualify it into a QSEv.⌋*()*

Notes:

- A SEv is able to have no filter chain associated to it.
- Several events can be assigned to a filter chain. All assigned events share the same settings of a filter. However, each assigned event has its own variable part for the filter (e.g., counter).

[SWS_IdsM_00904] ⌈Each filter shall reject a processed SEv in case the filter criteria are not met by dropping it.⌋*()*

[SWS_IdsM_00905] ⌈Otherwise, if a filter does not drop a SEv, the filter shall forward the currently processed SEv to the next filter in the chain.⌋*()*

[SWS_IdsM_00906] ⌈A filter shall be able to modify the SEvs counter according to their algorithm, if they are of type sampling or aggregation.⌋*()*

7.6 Filter Chain

Filter chains are configured using the SecXT model.

[SWS_IdsM_01001] Filter chain selection ⌈When a SEv is reported, the IdsM shall apply the filter chain that is mapped to it.⌋*(RS_IDS_00300, RS_IDS_00301)*

[SWS_IdsM_01002] Filter chain evaluation ⌈IdsM shall evaluate the filter chain after evaluating the reporting mode.⌋*(RS_IDS_00610)*

[SWS_IdsM_01003] Possible Filters ⌈Each filter chain may consist of the following filters:

- BlockState Filter
- Forward Every nth Filter
- Event Aggregation Filter
- Event Threshold Filter
- Event Rate Limitation
- Traffic Limitation

⌋*(RS_IDS_00300)*

Note: Each filter can be activated by aggregating the respective Filter object at the SecurityFilterChain object in the model.

[SWS_IdsM_01004] Filter chain order ⌈IdsM shall evaluate all activated filter in the order BlockState Filter, Forward-Every-nth Filter, Event Aggregation Filter, Event Threshold Filter.⌋*(RS_IDS_00300)*

[SWS_IdsM_01005] Dropping of SEvs ⌈If the evaluation of one filter leads to dropping the SEv, IdsM shall not evaluate any additional filter.⌋*(RS_IDS_00300)*

After successful evaluation of the configured filter chain, the security event is defined as qualified (QSEv).

The filters that compose a filter chain are categorized in the following groups:

- Blockers
- Sampling
- Aggregation
- Rate Limitation

Figure 7.5 shows the filter classification and their processing order.

- **Instance Specific Filters**: filter globally all SEvs that belong to a IdsM Instance: *Event Rate Limitation* and *Traffic Limitation*.
- **SEvID Specific Filters**: filter individually each SEv they are related to: *Reporting Mode*, *Block State*, *Forward Every Nth*, *Aggregation* and *Threshold*

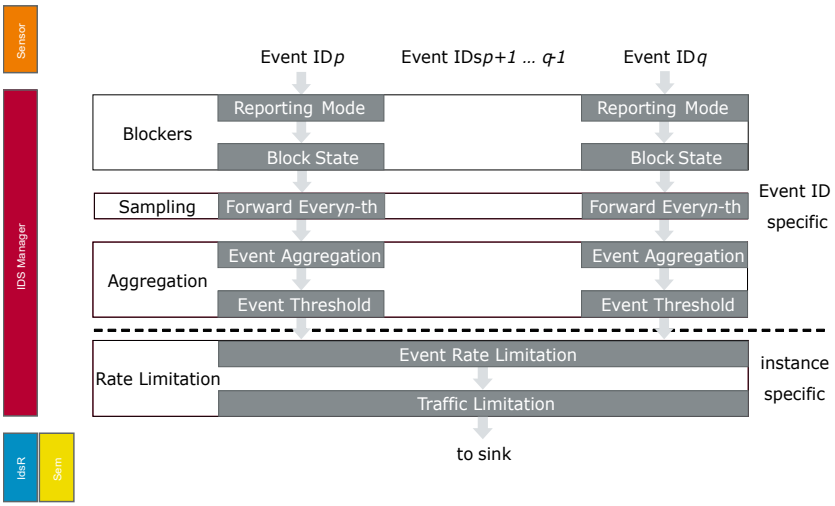

Figure 7.5: Filter categories and processing order

7.6.1 Blocker Filters

The blocker filters drop processed SEvs that do not match the filter criteria. These are filters specific to an instance of a SEv.

7.6.1.1 Reporting Mode Filter

The reporting mode filter enables the possibility to decide the detail of information of a SEv that is forwarded, bypass the filter chain or turn off the processing of certain SEv.

[SWS_IdsM_01010] ⌈The IdsM shall provide a **reporting mode** filter for each instance of SEv ID. This Filter is mandatory for each configured SEv.⌋*()*

[SWS_IdsM_01011] ⌈The reporting mode filter shall not part of a filter chain. It shall be directly linked to the respective SEv.⌋*()*

Note: A SEv does not have to be assigned to a filter chain.

[SWS_IdsM_01012] Reporting Mode ⌈IdsM shall determine the default reporting mode of each reported SEv from the IdsMReportingModeFilter.⌋*(RS_IDS_00310)*

[SWS_IdsM_01013] Reporting Mode Options ⌈IdsM shall handle a reported SEv depending on its reporting mode according to Table Table 7.1.⌋*(RS_IDS_00310)*

Reporting Mode Level	*Related Behavior*
OFF	IdsM shall discard the SEv without further processing.
BRIEF	If the SEv has been reported including context data, IdsM shall shall discard the context data from further processing, transmission, and storage.
DETAILED	If the SEv has been reported including context data, IdsM shall keep the context data for potential transmission or persisting of the QSEv.
BRIEF_BYPASSING_FILTERS	IdsM shall report or persist the SEv without context data without further applying of any filter chain.
DE-TAILED_BYPASSING_FILTERS	IdsM shall report or persist the SEv with context data (if provided by the sensor) without further applying of any filter chain.

Table 7.1: Reporting Mode Filter Values

Table Table 7.1 lists the possible values for the reporting mode filter IdsMReportingModeFilter. Depending on the literal chosen for the SEv the event will be filtered differently, and the context data will be dropped or passed to the next filter in the chain.

Note that the structure of the "Event Frame" is described in the Specification of Intrusion Detection System Protocol [4]. The reporting mode is independent of the *Configuration Features*: **Timestamp and Signature.**

7.6.1.2 Block State Filter

[SWS_IdsM_01020] ⌈The IdsM shall provide a **block state** filter. See IdsMBlockStateFilter.⌋*()*

[SWS_IdsM_01021] ⌈The **block state** filter shall represent a list of states in which the collection of the SEvs shall be blocked (the SEvs shall be dropped).⌋*()*

[SWS_IdsM_01022] ⌈The IdsM shall be inform, by the BswM module, about the current state with the callback service IdsM_BswM_StateChanged. This information shall be used when a block state filter is processed in the main function asynchronously.⌋*()*

[SWS_IdsM_01023] Block State Filter ⌈If IdsM evaluates the Block State Filter and the current block state equals one of the states referenced by IdsMBlockState, then IdsM shall drop the SEv.⌋*(RS_IDS_00320)*

[SWS_IdsM_01024] ⌈The **block state** filter shall forward the SEv to the next filter in the chain if the current state is not part of the list. In case this is the last filter in the chain, the filter forwards the QSEv to the sink.⌋*()*

Note: The possible States that can be contained in the filter are described in IdsMBlockState. The BswM reports the current IdsM state with the service IdsM_BswM_StateChanged using the symbolic name of the IdsM Block State Identifier IdsMBlockStateID.

The BswM has knowledge of the IdsM States available in the configuration by having a ECUC Reference to IdsMBlockState.

7.6.2 Sampling Filters

The sampling filters forward only certain events out of all incoming security events. These are filters specific to an instance of a SEv.

7.6.2.1 Forward Every Nth

[SWS_IdsM_01030] ⌈The IdsM shall provide **Forward Every Nth** filter.⌋*()*

[SWS_IdsM_01031] Sampling Filter ⌈If IdsM evaluates the sampling filter for a SEv, IdsM shall drop all the SEvs but every *nth*, where n is defined in IdsMNthParameter. Forwarding of SEvs starts with the first received SEv. Then every *nth* SEv is forwarded.⌋*(RS_IDS_00330)*

An implementation will typically maintain one counter that will be incremented when an SEv of given type is evaluated by the sampling filter. If the counter equals n the SEv is not dropped and the counter is reset to 0.

[SWS_IdsM_01032] Sampling Filter Initialization ⌈IdsM shall initialize the sampling filter for a SEv so that the first received SEv is forwarded.⌋*(RS_IDS_00330)* Example: IdsMNthParameter is set to 3 for a certain event type, then SEvs 1, 4, 7, ... will be forwarded by the IdsM (1 describing the first SEv reported after reset).

[SWS_IdsM_01033] ⌈The forwarding of the SEvs by the **Forward Every Nth filter** shall be done without modification of SEv data.⌋*()* e.g. Counter remains with the original value.

7.6.3 Aggregation Filters

All SEv of a given type occurring within a configured time interval are aggregated into one SEv with an additional counter information attached that indicates how often the event occurred in the time interval.

7.6.3.1 Event Aggregation Filter

[SWS_IdsM_01040] ⌈The IdsM shall provide an **aggregation** filter.⌋*()*

[SWS_IdsM_01041] Configuration of Event Aggregation Filter ⌈The parameter IdsMEventAggregationTimeInterval shall represent the duration of the interval during which SEvs of the given type shall be aggregated.⌋*(RS_IDS_00340)*

[SWS_IdsM_01042] ⌈The **aggregation** filter shall forward a SEv with the sum of the SEv's counters processed in an interval. Considering the configuration for a specific SEv, of the aggregation filter's time interval with value l_j for IdsMEventAggregationTimeInterval, the filter shall count the number of events of the same ID j received during a single aggregation time interval l_j⌋*()*

[SWS_IdsM_01043] No Event Forwarding During Interval ⌈The aggregation filter shall not forward (i.e., to the next filter) any incoming SEv during the aggregation interval.⌋*(RS_IDS_00340)*

At the end of each aggregation interval, the aggregation filter shall implement the following logic for each Security Event Type:

[SWS_IdsM_01044] End of Interval: No Event ⌈If no SEv of the same event type has been received by the aggregation filter in the past aggregation interval, no action shall be taken.⌋*(RS_IDS_00340)*

[SWS_IdsM_01045] End of Interval: One or More Events ⌈If one or more SEv of the same event type have been received by the aggregation filter in the past aggregation interval, a SEv shall be forwarded to the next filter in the chain.⌋*(RS_IDS_00340)*

[SWS_IdsM_01046] End of Interval: Count ⌈If the SEv is forwarded to the next filter in the filter chain, the count parameter of the SEv shall equal the sum of all count parameters of all SEvs of given event type processed by the aggregation filter in the past time interval.⌋*(RS_IDS_00340)*

[SWS_IdsM_01047] End of Interval: First Context Data ⌈If the SEv is forwarded to the next filter in the filter chain and if IdsMContextDataSourceSelector equals IDSM_FILTERS_CTX_USE_FIRST, then the context data shall equal the first context data of an SEv of given type that has been received at the aggregation filter in the past time interval.⌋*(RS_IDS_00340)*

[SWS_IdsM_01048] End of Interval: Last Context Data ⌈If the SEv is forwarded to the next filter in the filter chain and if IdsMContextDataSourceSelector equals IDSM_FILTERS_CTX_USE_LAST, then the context data shall equal the last context data of an SEv of given type that has been received at the aggregation filter in the past time interval.⌋*(RS_IDS_00340)*

[SWS_IdsM_01049] End of Interval: Timestamp ⌈If the SEv is forwarded to the next filter in the filter chain, the timestamp shall be taken from the same SEv from which the context data comes from (configured in IdsMContextDataSourceSelector).⌋*(RS_IDS_00340)*

[SWS_IdsM_01050] ⌈The time interval for each aggregation filter IdsMEventAggregationTimeInterval shall be a multiple of the main function period IdsMMainFunctionPeriod.⌋*()*

[SWS_IdsM_01051] ⌈The counting of the time interval for the aggregation filter `IdsMEventAggregationTimeInterval` shall start with the first call of the main function.⌋*()*

Figure 7.6 shows an example of the behavior of the aggregation filter for $EventId_{23}$, with *use last* context data source:

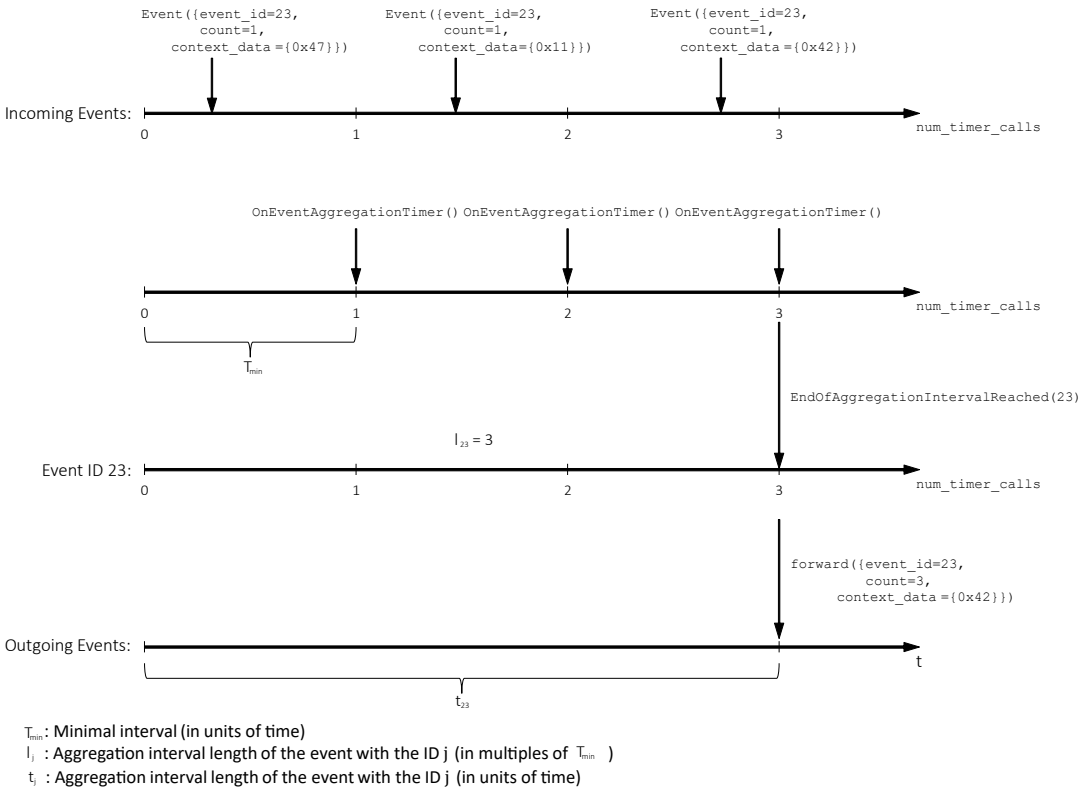

Figure 7.6: Example of aggregation filter

7.6.3.2 Event Threshold Filter

[SWS_IdsM_01060] ⌈The `IdsM` shall provide a **threshold** filter.⌋*()*

[SWS_IdsM_01061] Event Dropping Below Threshold ⌈The threshold filter shall drop an `SEv` of given type if the sum of count parameters of all `SEv`s of given type that were processed by the event threshold filter in the current threshold interval is smaller than the configured parameter `IdsMEventThresholdNumber`.⌋*(RS_IDS_00350)*

[SWS_IdsM_01062] Event Forwarding Above Threshold ⌈The threshold filter shall forward an `SEv` of given type if the sum of count parameters of all `SEv`s of given type that were processed by the event threshold filter in the current threshold interval is equal to or greater than the configured parameter `IdsMEventThresholdNumber`.⌋ *(RS_IDS_00350)*

Considering the configuration for a specific SEv, of the threshold filter l_j for IdsMEventThresholdTimeInterval, and a threshold number p for IdsMEventThresholdNumber, the filter shall count the incoming SEvs with the same ID j received during a single aggregation time interval l_j and drops the first $p-1$ events. All further incoming SEvs (equal or greater than p) shall be immediately forwarded until the end of the interval l_j.

[SWS_IdsM_01063] ⌈The counter of the events shall reset every time the threshold interval expires.⌋*()*

[SWS_IdsM_01064] ⌈The configured time interval for each threshold filter IdsMEventThresholdTimeInterval shall be a multiple of the IdsMMainFunctionPeriod.⌋*()*

[SWS_IdsM_01065] ⌈The counting of the time interval for the threshold filter IdsMEventThresholdTimeInterval shall start with the first call of the main function.⌋*()*

Figure 7.7 shows an example of the behavior of the aggregation filter for $\mathrm{EventId}_{47}$, with time interval equals 2 and threshold number equals 3:

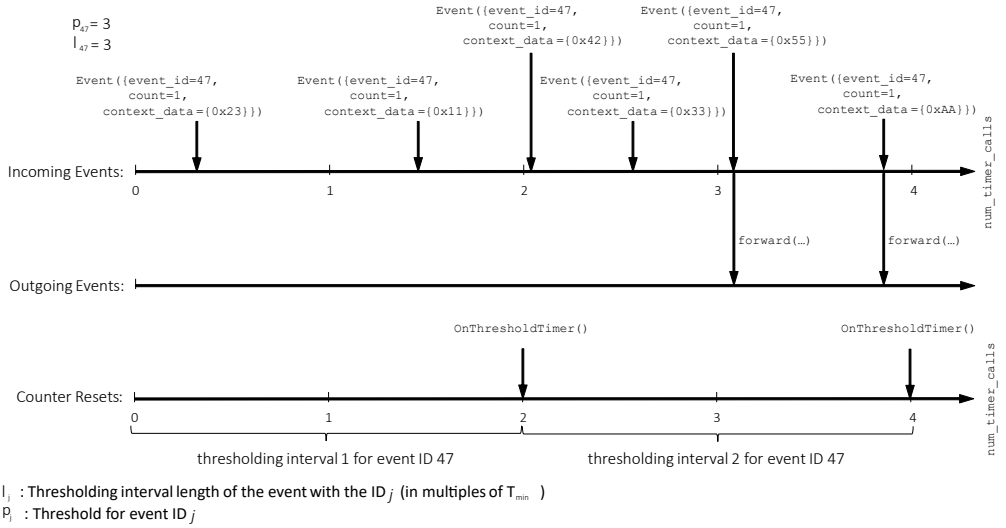

Figure 7.7: Example of threshold filter

7.6.4 Rate Limitation Filters

The rate limitation filters establish a maximum number of forwarded events in order to keep resources and avoid flooding of the system or network when reporting to the sinks. These are filters specific to an IdsM Instance.

[SWS_IdsM_01070] Rate and Traffic Limitation ⌈Before sending a QSEv to the IdsR, IdsM shall apply rate and traffic limitation that can lead to dropping the QSEv from transmission to the IdsR.⌋*(RS_IDS_00511)*

7.6.4.1 Event Rate Limitation

[SWS_IdsM_01080] ⌈The IdsM shall provide a **rate limitation** filter. This filter specifies a limit in number of SEvs and an interval in milliseconds.⌋*()*

[SWS_IdsM_01081] Rate Limitation ⌈IdsM shall drop an QSEv from transmission, if its transmission would cause the number of QSEvs transmitted in the current interval (specified in IdsMRateLimitationTimeInterval), to exceed the maximum number of transmission, configured in IdsMRateLimitationMaximumEvents.⌋*(RS_IDS_00511)*

[SWS_IdsM_01082] ⌈The time interval for the event rate limitation filter IdsMRateLimitationTimeInterval shall be a multiple of the main function period IdsMMainFunctionPeriod.⌋*()*

Note: This filter is not specific to a single SEv but it applies to all events handled by the current IdsM instance.

7.6.4.2 Traffic Limitation

[SWS_IdsM_01090] ⌈The IdsM shall provide a **traffic limitation** filter. This filter specifies a limit in bytes and an interval in milliseconds.⌋*()*

[SWS_IdsM_01091] Traffic Limitation ⌈IdsM shall drop an QSEv from transmission, if its transmission would cause the number of bytes transmitted in the current interval (specified in IdsMTrafficLimitationTimeInterval), to exceed the maximum number of bytes, configured in IdsMTrafficLimitationMaximumBytes.⌋*(RS_IDS_00511)*

[SWS_IdsM_01092] ⌈The time interval for the traffic limitation filter IdsMTrafficLimitationTimeInterval shall be a multiple of the main function period IdsMMainFunctionPeriod.⌋*()*

[SWS_IdsM_01093] ⌈The IdsM shall reset the byte counter to 0 when the interval IdsMTrafficLimitationTimeInterval expires.⌋*()*

[SWS_IdsM_01094] ⌈In case the number of bytes trying to be sent during a time period exceeds the maximum number of transmitted bytes IdsMTrafficLimitationMaximumBytes, The IdsM shall trigger the internal SEv IDSM_INTERNAL_EVENT_TRAFFIC_LIMITATION_EXCEEDED if configured.⌋*()* Please refer to [SWS_IdsM_91015] for the internal security events.

7.7 Timestamp

Timestamps are optional and can be provided to the IdsM in different ways and it shall be globally configured for all the QSEvs. The feature enables the ability to have a timestamp linked to a SEv.

The timestamp can be provided by a **smart sensor** or it shall be fetched from a chosen **timestamp origin**.

The origin of the timestamp can be chosen between: the one recorded by the application (custom timestamp) or an internal AUTOSAR timer from the Synchronized Time-Base Manager (StbM). Detailed timestamp information can be found in: Specification of Intrusion Detection System Protocol [4].

[SWS_IdsM_01100] ⌈The IdsM shall be able to add an additional **IDS Message Timestamp** field to the QSEv. The timestamp feature is optional and shall be activated and configured globally for all QSEvs with IdsMTimestampOption.⌋*()*

[SWS_IdsM_01101] ⌈In case the SEvs do not contain the optional information of a timestamp (the sensor does not include it when it reports a SEv to the IdsM), the IdsM shall request the timestamp information from the configured source in IdsMTimestampOption.⌋*()*

[SWS_IdsM_01103] ⌈The option **None** in IdsMTimestampOption shall disable the timestamp feature.⌋*()*

[SWS_IdsM_01104] ⌈The option **AUTOSAR** in IdsMTimestampOption shall enable the timestamp feature and determines that the source of the timestamp is the AUTOSAR StbM module by calling the function StbM_GetCurrentTime.⌋*()* See 9.2 for the interaction of the IdsM with the StbM as time stamp source.

The format of the timestamp to be added is specified in [4].

[SWS_IdsM_01105] ⌈The option **Custom** in IdsMTimestampOption shall enable the timestamp feature and determines that the source of the timestamp is provided by the application software.⌋*()*

[SWS_IdsM_01111] ⌈If the option **Custom** in IdsMTimestampOption is enabled, the IdsM shall use the Client Server Interface IdsM_CustomTimestamp with the operation Get, which the application shall implement, to request a timestamp from the SW-C via the *Require Port CustomTimestamp*.⌋*()* See 9.3 for the interaction of the IdsM with the SW-C as timestamp source.

[SWS_IdsM_01106] Timestamps are optional ⌈If the parameter IdsMTimestampOption is equals **None**, the IdsM shall not add a timestamp to a QSEv. The IdsM shall ignore timestamps provided via the timestamp parameter of the event reporting interface.⌋*(RS_IDS_00502)*

[SWS_IdsM_01107] Timestamps provided by the stack ⌈If IdsMTimestampOption is set to "'AUTOSAR"' and the SEv is reported without a timestamp parameter,

then IdsM shall add a timestamp from the StbM to the stored and transmitted QSEvs.⌋ *(RS_IDS_00503)*

[SWS_IdsM_01108] Timestamp provided via event reporting interface ⌈If the timestamp feature is enabled and the SEv is reported with a timestamp parameter via the services IdsM_SetSecurityEventWithTimestampCount or IdsM_SetSecurityEventWithTimestampCountContextData, then IdsM shall use this provided timestamp parameter for transmission or storage of the QSEv.⌋ *(RS_IDS_00503)*

[SWS_IdsM_01112] Timestamp not provided via event reporting interface ⌈If the timestamp feature is enabled and the SEv is reported without a timestamp parameter via the services IdsM_SetSecurityEventWithTimestampCount or IdsM_SetSecurityEventWithTimestampCountContextData, then IdsM shall set the *option bit for the time stamp* in protocol header to 0 before the transmission or storage of the QSEv.⌋ *(RS_IDS_00503)*

[SWS_IdsM_01109] Timestamp provided via application software ⌈If IdsMTimestampOption is set to "'Custom"', and the SEv is reported without a timestamp parameter, then IdsM shall request a timestamp from the application via the TimestampProvider callback and add the received timestamp to the QSEv.⌋ *(RS_IDS_00503)*

[SWS_IdsM_01110] Truncation of timestamp parameter ⌈If the SEv is reported with a timestamp parameter, then IdsM shall truncate this value by the 2 most-significant bits, i.e., only keep the 62 least-significant bits for further use.⌋ *(RS_IDS_00503)*

Please note that while the TimestampProvider API is specified, the integration and configuration of the TimestampProvider remains stack-vendor specific.

7.8 Reporting and Persistence of SEVs

Once the filter chain has processed the incoming SEvs, the resulting events from the processing filter chain are considered QSEvs.

Only QSEvs are further handled by the on-board IDS. These QSEvs can be either persisted in memory, sent to another ECU, or both depending on the configuration. The destination of a QSEv is called **data sink**.

7.8.1 Structure Of QSEVs

[SWS_IdsM_01200] ⌈The QSEvs shall have a defined structure independent of the sink it is being sent to. The components of a QSEv are listed in table Table 7.2. For further details of the IdsM Message structure refer to the [4, Specification of Intrusion Detection System Protocol].⌋ *()*

QSEv component	*Description*
Protocol Version and Header.	IdsM Protocol specific fields described in the [4, Specification of Intrusion Detection System Protocol]
IdsM Instance ID.	Specifies the IdsM Instance where the event originated from. This IdsM ID corresponds to the ECU of origin, as each ECU contains one *"Classic Platform IdsM instance"* at most.
Sensor Instance ID	Identifies the sensor instance in case there are several instances of the same sensor within an ECU.
Security Event Definition ID SEv Definition ID	Specifies the type of event. Every SEv type is identified with a unique ID.
Count	Represents the number of IdsM calls which have led to the current event. When an event is created, the counter shall be set to 1. Note: Filters like Event Aggregation may combine several events into a single one. For more details see the aggregation filter chapter:7.6.3.
Timestamp	Optional. Time of occurrence of a The SEv recorded by a smart sensor and forwarded to the IdsM Module.
Context Data	Contains additional binary data which semantics are opaque to the IDS. This is an optional field depending on the configuration and filtering of the corresponding SEvs.
Signature	Optional. Contains the signature of the SEv, calculated over the complete IdsM message.

Table 7.2: QSEv Structure

[SWS_IdsM_01201] ⌈The configuration of a SEv IdsMEvent shall contain a list of **data sinks** which are used for the resulting QSEv.⌋*()*

7.8.2 Propagation of QSEvs: IdsR Sink

[SWS_IdsM_01202] ⌈The IdsM shall provide the functionality for forwarding qualified on-board security events QSEvs to other ECUs via the PduR module.⌋*()*

Note: The transmission of QSEv to the backend (for use cases like off-board analysis) is supported by the IdsM concept but performed by another component (IdsR). Consult the [4, Specification of Intrusion Detection System Protocol] for the different message formats available for the transport of the event frame.

[SWS_IdsM_01203] QSEv transmission ⌈The IdsM shall be able to use the sink IdsR for the configured events. The IdsMSinkIdsR indicates that the corresponding QSEv shall be sent via PduR in a **IDS** Message to the communication network.⌋*(RS_IDS_00510)*

7.8.2.1 Authenticity of QSEvs: Signature

IdsM can optionally protect the authenticity of QSEvs using cryptographic signatures generated by the CSM in conjunction with the crypto stack. It can be used to ensure authenticity as well as to prove integrity of signed messages from the IdsM via all communication systems until reaching the Backend or SOC (End2End-Security).

[SWS_IdsM_01204] Signing QSEv ⌈The IdsM shall be able to attach a cryptographic signature, with the same data format, to each QSEv. The signature feature is optional and shall be activated or deactivated globally for all QSEvs with the presence of a configured Csm Job referenced by the IdsMCsmJobReference.⌋*(RS_IDS_00505)*

[SWS_IdsM_01205] ⌈The IdsM's Csm Job IdsMSignatureGenerateResultLength shall define the length in bytes of the signature calculated by the crypto services. It shall be configured when the signature feature is activated.⌋*()*

The IdsM's Csm Job IdsMCsmJobReference has two different types of signature processing: synchronous and asynchronous. This processing is configured in the Csm Job Primitive linked to the Csm Job, and determines the internal handling of the IdsM for the signature.

[SWS_IdsM_01206] ⌈In order to generate a signature by a Csm job IdsMCsmJobReference, the signature generation shall be triggered by calling the Csm function Csm_SignatureGenerate.⌋*()*

[SWS_IdsM_01207] ⌈If the signature is generated by a synchronous Csm job IdsMCsmJobReference, when the function Csm_SignatureGenerate returns, the signature shall be immediately available.⌋*()*

[SWS_IdsM_01208] ⌈If the signature is generated by an asynchronous Csm job IdsMCsmJobReference, the IdsM shall be informed about the generation of the signature by Csm via the Csm notification callback function IdsM_CsmNotification.⌋*()*

Note that the callback function IdsM_CsmNotification shall be configured in the Csm Module as a *Csm job primitive callback* for the Csm Job configured for the IdsM.

Since the signature is used for all QSEvs, i.e. independent of their sink configurations, the signature shall be generated before the QSEv is distributed to its configured sinks.

Over which data the signature shall be computed and how the signature shall be included in the IDS Message Structure, is specified in [4, Specification of Intrusion Detection System Protocol].

7.8.2.2 IDS Service Interface Options

The sensors coming from a SW-C or application have the option to transmit additional information to the IdsM. This option can be chosen individually per SEv under the parameter IdsMAdditionalParameterOption. It is possible to choose between having no additional information, report a counter and report a counter with timestamp.

[SWS_IdsM_01300] ⌈A SEvs reported by a SW-C shall define a maximum number of bytes for the transmission of the context data.IdsMEventMaxContextDataSize.⌋*()*

Note: a limitation for the number of bytes used between the IdsM and the RTE when forwarding context data of the corresponding security event, helps to avoid the waste of resources caused by the copying of data done by the RTE. With this limit, the size of data being copied can be tailored to the actual or similar amount of bytes that are being sent.

[SWS_IdsM_01301] No additional Interface Option ⌈The SW-C Service Port Interface shall not provide additional information other than the optional context data when the option **None** is configured in IdsMAdditionalParameterOption.⌋*()*

[SWS_IdsM_01302] Additional Interface Option: Count ⌈The SW-C Service Port Interface shall be extended by the parameter count when the option **Count** is configured in IdsMAdditionalParameterOption.⌋*()*

[SWS_IdsM_01303] Additional Interface Option: Count and timestamp ⌈The SW-C Service Port Interface shall be extended by the parameters count and timestamp when the option **CountTimestamp** is configured in IdsMAdditionalParameterOption.⌋*()*

7.8.2.3 Transmission Protocols

The IdsM shall calculate the **total size of the data to be transmitted**, depending on the size of the underlying **"Bus-PDU length"**, the IDS Message will be sent in different protocol types: interface (IF) or transport protocol (TP). Note that the total size of the data to be transmitted includes all mandatory and optional fields :

- IDS Message
  - Timestamp (optional)
  - Context Data (optional)
- IdsM Message Signature (optional)

[SWS_IdsM_01400] ⌈The IdsM shall send its data via a interface PDU (IF-PDU) if the complete IDS Message with its additional *IDS Message Signature*, if available, fits in a single Bus-PDU. Configured in: IdsMIfTxPdu.⌋*()*

[SWS_IdsM_01401] ⌈Otherwise, if the data does not fit in a single IF-PDU frame, it shall be send via transport protocol using TP-PDUs. Configured in: IdsMTpTxPdu.⌋*()*

Services for Reception

The IdsM does not receive data from the network, thus, it does not need the implementation of the reception services needed by the interface and transport protocols.

Services for Transmission

[SWS_IdsM_01498] ⌈After the IdsM has processed the SEvs, the resulting QSEvs which have passed the filtering and have the IdsMSinkIdsR configured, shall be transmitted using the service PduR_IdsMTransmit.⌋*()*

[SWS_IdsM_01499] ⌈IdsM shall not call PduR_IdsMTransmit again before IdsM_TpTxConfirmation or IdsM_TxConfirmation have been called.⌋*()*

[SWS_IdsM_01500] ⌈The IdsM shall receive the confirmation for the complete transmission of the IF upper layer Tx-Pdu by the PduR Module with the service IdsM_TxConfirmation.⌋*()*

When using the transport protocol (TP) for transmission of a segmented PDU, the following sequence shall be provided:

[SWS_IdsM_01501] ⌈The IdsM shall be able to transmit segmented PDUs with the service IdsM_CopyTxData. The function shall be called several times, each call to this function shall transmit a segment of the Tx-PDU, until it has been completely sent.⌋*()*

[SWS_IdsM_01502] ⌈The IdsM shall receive the confirmation of the transmission of a segmented PDU by the PduR Module with the service IdsM_TpTxConfirmation.⌋*()*

7.8.3 Storage of Events: Dem / Sem Sink

The DEM / Sem sink is not fully specified.

[SWS_IdsM_01600]{DRAFT} ⌈The IdsM shall provide a functionality for persisting on-board QSEvs, with their corresponding optional fields: context data and timestamp in Dem / Sem.⌋*(DRAFT)*

[SWS_IdsM_01601]{DRAFT} ⌈The IdsM shall be able to use the sink Dem / Sem for the configured events. The Sem sink indicates that the corresponding QSEv shall be stored in the Dem's user defined memory: Sem.⌋*(DRAFT)*

[SWS_IdsM_01602]{DRAFT} ⌈The data stored in Dem / Sem is the complete **IDS Message** with its mandatory and optional fields.⌋*(DRAFT)*

- IDS Message
  - Timestamp (optional)
  - Context Data (optional)
  - Signature (optional)

7.9 Persistence in NvM of Configuration

The value of the SEv's "Reporting Mode" IdsMReportingModeFilter is initially configured by the integrator during integration phase. However, it is possible to modify

its value during run-time via diagnostic services. For this reason, it is useful to persist the modified value once it has been changed.

[SWS_IdsM_01700] ⌈The IdsM shall be able to persist the parameter "Reporting Mode" of a SEv in the NvM.⌋*()*

[SWS_IdsM_01701] ⌈The write routine of the NvM block NvM_WriteBlock shall be triggered after the modification of a "Reporting Mode" value IdsMReportingModeFilter has been successfully changed by the diagnostic services.⌋*()* The modification of a "Reporting Mode" value is described in 7.10.1.

[SWS_IdsM_01702] ⌈If the persistence in the NvM block fails, the IdsM shall roll back the SEv's "Reporting Mode" IdsMReportingModeFilter, to the value before the diagnostic modification.⌋*()*

[SWS_IdsM_01703] ⌈The IdsM shall be able to read out the "Reporting Mode" IdsMReportingModeFilter persisted in the NvM for the corresponding SEvs handled by the IdsM instance.⌋*()*

[SWS_IdsM_01704] ⌈In case there are no NvM values available for the "Reporting Mode" IdsMReportingModeFilter of the SEvs, the configured values provided in the configuration tool shall be used.⌋*()*

[SWS_IdsM_01705] ⌈The NvM block descriptor referenced by IdsMNvmBlockDescriptor, shall be a block processed during NvM_ReadAll.⌋*()*

NvM_ReadAll is activated with the NvM option NvMSelectBlockForReadAll, and it checks if the RAM data is invalid (CRC) and restores the data from the NvM or load default values.

[SWS_IdsM_01706] ⌈The supported NvM RAM block name shall be IdsM_NvMRamBlockData.⌋*()*

[SWS_IdsM_01707] ⌈The supported NvM ROM block name shall be IdsM_NvMRomBlockData.⌋*()*

Notes: The NvM should be already initialize before the IdsM is initialized. The Rom block is a basic storage object that provides default data in case of an empty or damaged NV block. It should be filled in with the default values of the configuration.

7.10 Diagnostics for SEvs

[SWS_IdsM_01800] ⌈The diagnostic handling feature shall be optional. It shall be activated or deactivated with the parameter IdsMDiagnosticSupport.⌋*()*

The diagnostic handling feature includes: reconfiguration of SEvs and reading of SEvs' parameters.

7.10.1 Reconfiguration of SEvs

[SWS_IdsM_01900] ⌈The "Reporting Mode" IdsMReportingModeFilter of a SEv shall be modifiable during run-time via the diagnostic services of the Dcm.⌋*()*

[SWS_IdsM_01901] ⌈The service IdsM_Dcm_SetReportingMode_Start called by the Dcm module shall enable the IdsM to modify the reporting mode IdsMReportingModeFilter of a specific SEv. This service shall trigger the routine execution to modify the current reporting mode, and shall contain the new reporting mode value to be set.⌋*()*

Note that immediately after modifying a reporting mode, the new mode will be persisted in NvM if the feature is active 7.9.

[SWS_IdsM_01902] ⌈In case the reporting mode parameter in IdsM_Dcm_SetReportingMode_Start is invalid, this IdsM service shall return DCM_E_REQUESTOUTOFRANGE as its Dcm negative response and the function shall return E_NOT_OK.⌋*()*

[SWS_IdsM_01903] ⌈In case the *Security Event Definition Id* used for the call IdsM_Dcm_SetReportingMode_Start is invalid, this IdsM service shall return DCM_E_REQUESTOUTOFRANGE as its Dcm negative response and the function shall return E_NOT_OK.⌋*()*

[SWS_IdsM_01904] ⌈In case the request to NvM to persist the new reporting mode fails, this IdsM service shall return DCM_E_GENERALPROGRAMMINGFAILURE as its Dcm negative response and the function shall return E_NOT_OK.⌋*()*

[SWS_IdsM_01905] ⌈In case the request to NvM to persist the new reporting mode fails, this IdsM service shall roll back to the previously configured reporting mode.⌋*()*

7.10.2 Reading of SEvs Reporting Mode

[SWS_IdsM_02000] ⌈The "Reporting Mode" IdsMReportingModeFilter of a SEv shall be readable via the diagnostic services of the Dcm.⌋*()*

In order to read out the "Reporting mode of a specific SEv the following diagnostic sequence shall be followed:

[SWS_IdsM_02001] ⌈The service IdsM_Dcm_GetReportingMode_Start called by the Dcm module shall trigger the IdsM's routine execution to request the current reporting mode of a specific SEv.⌋*()*

[SWS_IdsM_02002] ⌈The service IdsM_Dcm_GetReportingMode_RequestResults called by the Dcm module shall allow the IdsM to provide the routine results and reporting mode for the requested security event via a result pointer.⌋*()*

7.11 Error Classification

7.11.1 Development Errors

[SWS_IdsM_02003] ⌈

Type of error	*Related error code*	*Error value*
API function called with an invalid event identifier.	IDSM_E_PARAM_INVALID	0x0A
API function called with a NULL pointer parameter.	IDSM_E_PARAM_POINTER	0x0B
API function called with an invalid data size parameter.	IDSM_E_PARAM_LENGTH	0x0C
API function called before IdsM has been fully initialized.	IDSM_E_UNINIT	0x0D
The service IdsM_Init is called while the module is already initialized.	IDSM_E_ALREADY_INITIALIZED	0x0E

⌋()

7.11.2 Runtime Errors

The IdsM module does not define runtime errors.

7.11.3 Transient Faults

The IdsM module does not define transient errors.

7.11.4 Production Errors

The IdsM module does not define production errors.

7.11.5 Extended Production Errors

The IdsM module does not define extended production errors.

7.12 Error Detection and Notification

For details about error detection and notification of BSW modules refer to the chapter 7.2 “Error Handling” in [3, SWS_BSWGeneral].

7.12.1 Api Parameter Checking

The IdsM module reports the development error IDSM_E_PARAM_POINTER when a NULL_PTR is not accepted as an argument to a service or callback function. The exact behavior is specified in [SWS_BSW_00050] and [SWS_BSW_00212].

[SWS_IdsM_02101] ⌈With development error detection IdsMDevErrorDetect enabled, the IdsM module shall check the parameter securityEventId of the function IdsM_SetSecurityEvent against the configured security events, and shall report the development error IDSM_E_PARAM_INVALID when an unknown event ID is provided by the service call. An unknown event is an event that has not been configured in IdsMEvent.⌋*()*

[SWS_IdsM_02102] ⌈With development error detection IdsMDevErrorDetect enabled, the IdsM module shall check the parameter securityEventId of the function IdsM_SetSecurityEventWithContextData against the configured security events, and shall report the development error IDSM_E_PARAM_INVALID when an unknown event ID is provided by the service call. An unknown event is an event that has not been configured in IdsMEvent.⌋*()*

[SWS_IdsM_02103] ⌈With development error detection IdsMDevErrorDetect enabled, the IdsM module shall check the parameters securityEventId and count of the function IdsM_SetSecurityEventWithCount. The development error IDSM_E_PARAM_INVALID shall be reported when an unknown event ID is provided by the service call or the passed count is equal to 0.⌋*()*

[SWS_IdsM_02104] ⌈With development error detection IdsMDevErrorDetect enabled, the IdsM module shall check the parameters securityEventId and count of the function IdsM_SetSecurityEventWithCountContextData. The development error IDSM_E_PARAM_INVALID shall be reported when an unknown event ID is provided by the service call or the passed count is equal to 0.⌋*()*

[SWS_IdsM_02105] ⌈With development error detection IdsMDevErrorDetect enabled, the IdsM module shall check the parameters securityEventId and count of the function IdsM_SetSecurityEventWithTimestampCount. The development error IDSM_E_PARAM_INVALID shall be reported when an unknown event ID is provided by the service call or the passed count is equal to 0.⌋*()*

[SWS_IdsM_02106] ⌈With development error detection IdsMDevErrorDetect enabled, the IdsM module shall check the parameters securityEventId and count of the function IdsM_SetSecurityEventWithTimestampCountContextData. The development error IDSM_E_PARAM_INVALID shall be reported when an unknown event ID is provided by the service call or the passed count is equal to 0.⌋*()*

[SWS_IdsM_02107] ⌈With development error detection IdsMDevErrorDetect enabled, the IdsM module shall check the parameter contextDataSize of the function

IdsM_SetSecurityEventWithContextData, and shall report the development error IDSM_E_PARAM_LENGTH when the value of the parameter exceeds the maximum configured context data buffer size. The maximum context data buffer size results from the largest configured IdsMContextDataBufferSize.⌋()

[SWS_IdsM_02108] ⌈With development error detection IdsMDevErrorDetect enabled, the IdsM module shall check the parameter contextDataSize of the function IdsM_SetSecurityEventWithCountContextData, and shall report the development error IDSM_E_PARAM_LENGTH when the value of the parameter exceeds the maximum configured context data buffer size. The maximum context data buffer size results from the largest configured IdsMContextDataBufferSize.⌋()

[SWS_IdsM_02109] ⌈With development error detection IdsMDevErrorDetect enabled, the IdsM module shall check the parameter contextDataSize of the function IdsM_SetSecurityEventWithTimestampCountContextData, and shall report the development error IDSM_E_PARAM_LENGTH when the value of the parameter exceeds the maximum configured context data buffer size. The maximum context data buffer size results from the largest configured IdsMContextDataBufferSize.⌋()

Notice that the API is called with the symbolic name of the configured SEv.

8 API specification

8.1 Imported Types

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

[SWS_IdsM_91022] ⌈

Module	*Header File*	*Imported Type*
ComStack_Types	ComStack_Types.h	BufReq_ReturnType
	ComStack_Types.h	PduIdType
	ComStack_Types.h	PduInfoType
	ComStack_Types.h	PduLengthType
	ComStack_Types.h	RetryInfoType
	ComStack_Types.h	TpDataStateType
Dcm	Rte_Dcm_Type.h	Dcm_NegativeResponseCodeType
	Rte_Dcm_Type.h	Dcm_OpStatusType
StbM	Rte_StbM_Type.h	StbM_SynchronizedTimeBaseType
	Rte_StbM_Type.h	StbM_TimeBaseStatusType
	Rte_StbM_Type.h	StbM_TimeStampType
	Rte_StbM_Type.h	StbM_UserDataType
Std	Std_Types.h	Std_ReturnType
	Std_Types.h	Std_VersionInfoType

⌋()

8.2 Type Definitions

8.2.1 IdsM_ConfigType

[SWS_IdsM_91012] ⌈

Name	IdsM_ConfigType	
Kind	Structure	
Elements	implementation specific	
	Type	–
	Comment	–
Description	Configuration data structure of IdsM module.	
Available via	IdsM.h	

⌋()

8.2.2 IdsM_Filters_BlockStateType

[SWS_IdsM_91017] ⌈

Name	IdsM_Filters_BlockStateType		
Kind	Type		
Derived from	uint16		
Range	0..65535	–	–
Description	Data type used for "Block State" filter values (bit masks)		
Available via	IdsM_Filters_Types.h		

⌋()

8.2.3 IdsM_Filters_ReportingModeType

[SWS_IdsM_91013] ⌈

Name	IdsM_Filters_ReportingModeType		
Kind	Type		
Derived from	uint8		
Range	IDSM_REPORTING_MODE_OFF	0x00	Off: Event is not reported
	IDSM_REPORTING_MODE_ BRIEF	0x01	Brief: Event is reported without context data
	IDSM_REPORTING_MODE_ DETAILED	0x02	Detailed: Event is reported including context data
	IDSM_REPORTING_MODE_ BRIEF_BYPASSING_FILTERS	0x03	Brief, bypassing filters: Event is reported unfiltered without context data
	IDSM_REPORTING_MODE_ DETAILED_BYPASSING_ FILTERS	0x04	Detailed, bypassing filters: Event is reported unfiltered including context data
	IDSM_REPORTING_MODE_ INVALID	0xFF	Invalid reporting mode
Description	Reporting modes used by the reporting mode filter		
Available via	IdsM_Types.h		

⌋()

8.2.4 IdsM_TimestampType

[SWS_IdsM_91014] ⌈

Name	IdsM_TimestampType
Kind	Type
Derived from	uint64
Description	Data type for IdsM timestamps
Available via	IdsM_Types.h

⌋()

8.3 Function Definitions

8.3.1 IdsM_Init

[SWS_IdsM_91001] ⌈

Service Name	IdsM_Init	

Syntax	`void IdsM_Init (`
	`  const IdsM_ConfigType* configPtr`
	`)`	

Service ID [hex]	0x00	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	configPtr	Component configuration structure
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Service to initialize the module IdsM. It initializes all variables and sets the module state to initialized.	
Available via	IdsM.h	

⌋()

8.3.2 IdsM_GetVersionInfo

[SWS_IdsM_91004] ⌈

Service Name	IdsM_GetVersionInfo

Syntax	`void IdsM_GetVersionInfo (`
	`  const Std_VersionInfoType* versionInfo`
	`)`

Service ID [hex]	0x01

▽

△

Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	None	
Parameters (inout)	None	
Parameters (out)	versionInfo	Pointer to where to store the version information. Parameter must not be NULL.
Return value	None	
Description	Returns version information, vendor ID and AUTOSAR module ID of the component.	
Available via	IdsM.h	

⌋()

8.3.3 IdsM_MainFunction

[SWS_IdsM_91000] ⌈

Service Name	IdsM_MainFunction
Syntax	`void IdsM_MainFunction ( void )`
Service ID [hex]	0x02
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Parameters (in)	None
Parameters (inout)	None
Parameters (out)	None
Return value	None
Description	This function is called periodically. It processes security events asynchronously which are queued during API function calls.
Available via	IdsM.h

⌋()

8.3.4 IdsM_CopyTxData

[SWS_IdsM_91010] ⌈

Service Name	IdsM_CopyTxData

▽

△

Syntax	`BufReq_ReturnType IdsM_CopyTxData ( PduIdType id, const PduInfoType* info, const RetryInfoType* retry, PduLengthType* availableDataPtr )`	
Service ID [hex]	0x43	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	id	Identification of the transmitted I-PDU.
	info	Provides the destination buffer (SduDataPtr) and the number of bytes to be copied (SduLength). If not enough transmit data is available, no data is copied by the upper layer module and BUFREQ_E_BUSY is returned. The lower layer module may retry the call. An SduLength of 0 can be used to indicate state changes in the retry parameter or to query the current amount of available data in the upper layer module. In this case, the Sdu DataPtr may be a NULL_PTR.
	retry	This parameter is used to acknowledge transmitted data or to retransmit data after transmission problems. If the retry parameter is a NULL_PTR, it indicates that the transmit data can be removed from the buffer immediately after it has been copied. Otherwise, the retry parameter must point to a valid RetryInfoType element. If TpDataState indicates TP_CONFPENDING, the previously copied data must remain in the TP buffer to be available for error recovery. TP_DATACONF indicates that all data that has been copied before this call is confirmed and can be removed from the TP buffer. Data copied by this API call is excluded and will be confirmed later. TP_DATARETRY indicates that this API call shall copy previously copied data in order to recover from an error. In this case TxTpDataCnt specifies the offset in bytes from the current data copy position.
Parameters (inout)	None	
Parameters (out)	availableDataPtr	Indicates the remaining number of bytes that are available in the upper layer module's Tx buffer. availableDataPtr can be used by TP modules that support dynamic payload lengths (e.g. FrIsoTp) to determine the size of the following CFs.
Return value	BufReq_ReturnType	BUFREQ_OK: Data has been copied to the transmit buffer completely as requested. BUFREQ_E_BUSY: Request could not be fulfilled, because the required amount of Tx data is not available. The lower layer module may retry this call later on. No data has been copied. BUFREQ_E_NOT_OK: Data has not been copied. Request failed.
Description	This function is called to acquire the transmit data of an I-PDU segment (N-PDU). Each call to this function provides the next part of the I-PDU data unless retry->TpDataState is TP_DATARETRY. In this case the function restarts to copy the data beginning at the offset from the current position indicated by retry->TxTpDataCnt. The size of the remaining data is written to the position indicated by availableDataPtr.	
Available via	IdsM_Cbk.h	

⌋()

8.3.5 IdsM_SetSecurityEvent

[SWS_IdsM_91002] ⌈

Service Name	IdsM_SetSecurityEvent	
Syntax	`void IdsM_SetSecurityEvent ( IdsM_SecurityEventIdType securityEventId )`	
Service ID [hex]	0x03	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	securityEventId	Security Event ID
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	This API is the application interface to report security events to the IdsM.	
Available via	IdsM.h	

⌋()

8.3.6 IdsM_SetSecurityEventWithContextData

[SWS_IdsM_91003] ⌈

Service Name	IdsM_SetSecurityEventWithContextData	
Syntax	`void IdsM_SetSecurityEventWithContextData ( IdsM_SecurityEventIdType securityEventId, const uint8* contextData, uint16 contextDataSize )`	
Service ID [hex]	0x04	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	securityEventId	Security Event ID
	contextData	Pointer to optional context data. Use NULL_PTR if no context data is available.
	contextDataSize	Size of context data
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	This API is the application interface to report security events with context data to the IdsM.	
Available via	IdsM.h	

⌋()

8.3.7 IdsM_SetSecurityEventWithCount

[SWS_IdsM_91018] ⌈

Service Name	IdsM_SetSecurityEventWithCount	
Syntax	`void IdsM_SetSecurityEventWithCount ( IdsM_SecurityEventIdType securityEventId, uint16 count )`	
Service ID [hex]	0x05	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	securityEventId	Security event ID
	count	Count value which is used as the start value for the security event.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	This API is the application interface for Smart Sensors to report security events with a count value to the IdsM.	
Available via	IdsM.h	

⌋()

8.3.8 IdsM_SetSecurityEventWithCountContextData

[SWS_IdsM_91019] ⌈

Service Name	IdsM_SetSecurityEventWithCountContextData	
Syntax	`void IdsM_SetSecurityEventWithCountContextData ( IdsM_SecurityEventIdType securityEventId, uint16 count, const uint8* contextData, uint16 contextDataSize )`	
Service ID [hex]	0x06	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	securityEventId	Security event ID
	count	Count value which is used as the start value for the security event.
	contextData	Pointer to optional context data. Use NULL_PTR if no context data is available.
	contextDataSize	Size of context data
Parameters (inout)	None	
Parameters (out)	None	

▽

△

Return value	None
Description	This API is the application interface for Smart Sensors to report security events with a count value and context data to the IdsM.
Available via	IdsM.h

⌋()

8.3.9 IdsM_SetSecurityEventWithTimestampCount

[SWS_IdsM_91020] ⌈

Service Name	IdsM_SetSecurityEventWithTimestampCount	
Syntax	`void IdsM_SetSecurityEventWithTimestampCount ( IdsM_SecurityEventIdType securityEventId, IdsM_TimestampType timestamp, uint16 count )`	
Service ID [hex]	0x07	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	securityEventId	Security event ID
	timestamp	Timestamp used for time reference of the security event.
	count	Count value which is used as the start value for the security event.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	This API is the application interface for Smart Sensors to report security events with a timestamp and a count value to the IdsM.	
Available via	IdsM.h	

⌋()

8.3.10 IdsM_SetSecurityEventWithTimestampCountContextData

[SWS_IdsM_91021] ⌈

Service Name	IdsM_SetSecurityEventWithTimestampCountContextData

▽

△

Syntax	`void IdsM_SetSecurityEventWithTimestampCountContextData ( IdsM_SecurityEventIdType securityEventId, IdsM_TimestampType timestamp, uint16 count, const uint8* contextData, uint16 contextDataSize )`	
Service ID [hex]	0x08	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	securityEventId	Security event ID
	timestamp	Timestamp used for time reference of the security event.
	count	Count value which is used as the start value for the security event.
	contextData	Pointer to optional context data. Use NULL_PTR if no context data is available.
	contextDataSize	Size of context data
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	This API is the application interface for Smart Sensors to report security events with a timestamp, a count value and context data to the IdsM.	
Available via	IdsM.h	

⌋()

8.4 Callback Notifications

This is a list of functions provided for other modules.

8.4.1 IdsM_BswM_StateChanged

[SWS_IdsM_91005] ⌈

Service Name	IdsM_BswM_StateChanged	
Syntax	`void IdsM_BswM_StateChanged ( IdsM_Filters_BlockStateType state )`	
Service ID [hex]	0x0F	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	state	Current ECU state
Parameters (inout)	None	

▽

△

Parameters (out)	None
Return value	None
Description	This callback function is invoked by the BswM to indicate ECU state changes.
Available via	IdsM_Cbk.h

⌋()

8.4.2 IdsM_TpTxConfirmation

[SWS_IdsM_91011] ⌈

Service Name	IdsM_TpTxConfirmation	
Syntax	`void IdsM_TpTxConfirmation ( PduIdType id, Std_ReturnType result )`	
Service ID [hex]	0x48	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	id	Identification of the transmitted I-PDU.
	result	Result of the transmission of the I-PDU.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	This function is called after the I-PDU has been transmitted on its network, the result indicates whether the transmission was successful or not.	
Available via	IdsM_Cbk.h	

⌋()

8.4.3 IdsM_TxConfirmation

[SWS_IdsM_91009] ⌈

Service Name	IdsM_TxConfirmation
Syntax	`void IdsM_TxConfirmation ( PduIdType TxPduId, Std_ReturnType result )`
Service ID [hex]	0x40
Sync/Async	Synchronous

▽

△

Reentrancy	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in)	TxPduId	ID of the PDU that has been transmitted.
	result	E_OK: The PDU was transmitted. E_NOT_OK: Transmission of the PDU failed.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	The lower layer communication interface module confirms the transmission of a PDU, or the failure to transmit a PDU.	
Available via	IdsM_Cbk.h	

⌋()

8.4.4 IdsM_Dcm_GetReportingMode_RequestResults

[SWS_IdsM_91007] ⌈

Service Name	IdsM_Dcm_GetReportingMode_RequestResults	
Syntax	`Std_ReturnType IdsM_Dcm_GetReportingMode_RequestResults ( Dcm_OpStatusType OpStatus , uint8* Out_ReportingMode, Dcm_NegativeResponseCodeType* ErrorCode )`	
Service ID [hex]	0x0D	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	OpStatus	The operation status
Parameters (inout)	None	
Parameters (out)	Out_ReportingMode	The reporting mode for the requested Security Event
	ErrorCode	Negative Response code, in case of an failure
Return value	Std_ReturnType	E_OK: The operation is finished DCM_E_PENDING: The operation is not yet finished E_NOT_OK The operation has failed. A concrete NRC shall be set, otherwise the DCM sends NRC 0x22
Description	This function is a request from DCM to the IdsM to read the routine results triggered by function IdsM_Dcm_GetReportingMode_Start().	
Available via	IdsM_Cbk.h	

⌋()

8.4.5 IdsM_Dcm_GetReportingMode_Start

[SWS_IdsM_91006] ⌈

Service Name	IdsM_Dcm_GetReportingMode_Start	
Syntax	`Std_ReturnType IdsM_Dcm_GetReportingMode_Start ( uint16 In_SecurityEventId, uint8 In_SensorInstanceId, Dcm_OpStatusType OpStatus, Dcm_NegativeResponseCodeType* ErrorCode )`	
Service ID [hex]	0x0C	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	In_SecurityEventId	External ID of the Security Event from whom the reporting mode shall be returned
	In_SensorInstanceId	ID of the sensor instance of the security event
	OpStatus	The operation status
Parameters (inout)	None	
Parameters (out)	ErrorCode	Negative Response code, in case of an failure
Return value	Std_ReturnType	E_OK: The operation is finished E_NOT_OK: The operation has failed. A concrete NRC shall be set, otherwise the DCM sends NRC 0x22 DCM_E_PENDING: The operation is not yet finished
Description	This function is a request from DCM to the IdsM to start the routine execution to request the current reporting mode of a specific Security Event ID.	
Available via	IdsM_Cbk.h	

⌋()

8.4.6 IdsM_Dcm_SetReportingMode_Start

[SWS_IdsM_91008] ⌈

Service Name	IdsM_Dcm_SetReportingMode_Start	
Syntax	`Std_ReturnType IdsM_Dcm_SetReportingMode_Start ( uint16 In_SecurityEventId, uint8 In_SensorInstanceId, uint8 In_ReportingMode, Dcm_OpStatusType OpStatus, Dcm_NegativeResponseCodeType* ErrorCode )`	
Service ID [hex]	0x0E	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	In_SecurityEventId	External ID of the Security Event from whom the reporting mode shall be altered
	In_SensorInstanceId	ID of the sensor instance of the security event
	In_ReportingMode	Reporting Mode which shall be stored
	OpStatus	The operation status

▽

△

Parameters (inout)	None	
Parameters (out)	ErrorCode	Negative Response code, in case of an failure
Return value	Std_ReturnType	E_OK The operation is finished DCM_E_PENDING The operation is not yet finished E_NOT_OK The operation has failed. A concrete NRC shall be set, otherwise the DCM sends NRC 0x22
Description	This function is a request from DCM to the IdsM to start the routine execution to set the reporting mode of a specific Security Event ID.	
Available via	IdsM_Cbk.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.

8.6 Expected Interfaces

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

8.6.1 Mandatory Interfaces

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

[SWS_IdsM_91023] ⌈

API Function	***Header File***	***Description***
There are no mandatory interfaces.		

⌋()

8.6.2 Optional Interfaces

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

[SWS_IdsM_91024] ⌈

API Function	Header File	Description
StbM_GetCurrentTime	StbM.h	Returns a time value (Local Time Base derived from Global Time Base) in standard format. Note: This API shall be called with locked interrupts / within an Exclusive Area to prevent interruption (i.e., the risk that the time stamp is outdated on return of the function call).

⌋()

8.7 Service Interfaces

8.7.1 Client-Server Interfaces

8.7.2 IdsM_IdsMService

[SWS_IdsM_91027] ⌈

Name	IdsMService_{EventName}		
Comment	Interface to report security events to the IdsM. Depending on the configuration of the event, thus on the number and type of parameters passed to the IdsM about the event, a different operation shall be used.		
IsService	true		
Variation	({ecuc(IdsM/IdsMConfiguration/IdsMEvent/IdsMExternalEventId)} is in range 0x8000 - 0xFFFE) ({ecuc(IdsM/IdsMConfiguration/IdsMEvent/IdsMServiceInterfaceOptions)} EXISTS IdsM/IdsMConfiguration/IdsMEvent/IdsMServiceInterfaceOptions/IdsMAdditionalParameterOption == None EventName = {ecuc(IdsM/IdsMConfiguration/IdsMEvent.SHORT-NAME)}		
Possible Errors	–	–	–

Operation	SetSecurityEvent
Comment	This function shall report security events to the IdsM only with the SecurityEventId
Variation	({ecuc(IdsM/IdsMConfiguration/IdsMEvent/IdsMServiceInterfaceOptions/IdsMAdditionalParameterOption)} == None)({ecuc(IdsM/IdsMConfiguration/IdsMEvent/IdsMServiceInterfaceOptions/IdsMEventMaxContextDataSize)} == 0)
Possible Errors	–

Operation	SetSecurityEventWithContextData	
Comment	This function shall report a security event with context data	
Variation	({ecuc(IdsM/IdsMConfiguration/IdsMEvent/IdsMServiceInterfaceOptions/IdsMAdditionalParameterOption)} == None)({ecuc(IdsM/IdsMConfiguration/IdsMEvent/IdsMServiceInterfaceOptions/IdsMEventMaxContextDataSize)} > 0)	
	contextData	
	Type	IdsM _{EventName}_ContextDataType

▽

△

	Direction	IN
	Comment	Pointer to optional context data. Use NULL_PTR if no context data is available.
	Variation	EventName = {ecuc(IdsM/IdsMConfiguration/IdsMEvent.SHORT-NAME)}
	contextDataSize	
	Type	uint16
	Direction	IN
	Comment	Size of context data, must be in range of 0...<Size of maximum configured context data buffer>
	Variation	–
Possible Errors	–	

⌋()

8.7.3 IdsM_SmartSensorService

[SWS_IdsM_91028] ⌈

Name	IdsMSmartSensorService_{EventName}		
Comment	Interface to report security events to the IdsM used by a smart sensor. Depending on the configuration of the event, thus on the number and type of parameters passed to the IdsM about the event, a different operation shall be used.		
IsService	true		
Variation	({ecuc(IdsM/IdsMConfiguration/IdsMEvent/IdsMExternalEventId)} is in range 0x8000 - 0xFFFE)({ecuc(IdsM/IdsMConfiguration/IdsMEvent/IdsMServiceInterfaceOptions)} EXISTSIdsM/IdsMConfiguration/IdsMEvent/IdsMServiceInterfaceOptions/IdsMAdditionalParameterOption != None EventName = {ecuc(IdsM/IdsMConfiguration/IdsMEvent.SHORT-NAME)}		
Possible Errors	–	–	–

Operation	SetSecurityEventWithCount	
Comment	This function shall be used by smart sensors to report security events with a count value to the IdsM.	
Variation	({ecuc(IdsM/IdsMConfiguration/IdsMEvent/IdsMServiceInterfaceOptions/IdsMAdditionalParameterOption == Count)({ecuc(IdsM/IdsMConfiguration/IdsMEvent/IdsMServiceInterfaceOptions/IdsMEventMaxContextDataSize == 0)	
Parameters	count	
	Type	uint16
	Direction	IN
	Comment	Count value which is used as the start value for the security event, must be in range of 1...65535
	Variation	–
Possible Errors	–	

Operation	SetSecurityEventWithCountContextData	
Comment	This function shall be used by smart sensors to report a security event with count value and context data to the IdsM.	
Variation	({ecuc(IdsM/IdsMConfiguration/IdsMEvent/IdsMServiceInterfaceOptions/IdsMAdditional ParameterOption = Count)({ecuc(IdsM/IdsMConfiguration/IdsMEvent/IdsMServiceInterface Options/IdsMEventMaxContextDataSize > 0)	
Parameters	count	
	Type	uint16
	Direction	IN
	Comment	Count value which is used as the start value for the security event, must be in range of 1...65535
	Variation	–
	contextData	
	Type	IdsM _{EventName}_ContextDataType
	Direction	IN
	Comment	Pointer to optional context data. Use NULL_PTR if no context data is available.
	Variation	EventName = {ecuc(IdsM/IdsMConfiguration/IdsMEvent.SHORT-NAME)}
	contextDataSize	
	Type	uint16
	Direction	IN
	Comment	Size of context data, must be in range of 0...<Size of maximum configured context data buffer>.
	Variation	–
Possible Errors	–	

Operation	SetSecurityEventWithTimestampCount	
Comment	This function shall be used by smart sensors to report a security event with timestamp and count value to the IdsM.	
Variation	({ecuc(IdsM/IdsMConfiguration/IdsMEvent/IdsMServiceInterfaceOptions/IdsMAdditional ParameterOption == CountTimestamp)({ecuc(IdsM/IdsMConfiguration/IdsMEvent/IdsMService InterfaceOptions/IdsMEventMaxContextDataSize == 0)	
Parameters	timestamp	
	Type	uint64
	Direction	IN
	Comment	Timestamp used for time reference of the security event, must be in range of 0...(2^62 - 1)
	Variation	–
	count	
	Type	uint16
	Direction	IN
	Comment	Count value which is used as the start value for the security event, must be in range of 1...65535
	Variation	–
Possible Errors	–	

Operation	SetSecurityEventWithTimestampCountContextData	
Comment	This function shall be used by smart sensors to report a security event with timestamp, count value and context data.	
Variation	({ecuc(IdsM/IdsMConfiguration/IdsMEvent/IdsMServiceInterfaceOptions/IdsMAdditional ParameterOption == CountTimestamp)({ecuc(IdsM/IdsMConfiguration/IdsMEvent/IdsMService InterfaceOptions/IdsMEventMaxContextDataSize > 0)	
Parameters	timestamp	
	Type	uint64
	Direction	IN
	Comment	Timestamp used for time reference of the security event, must be in range of 0...(2^62 - 1)
	Variation	–
	count	
	Type	uint16
	Direction	IN
	Comment	Count value which is used as the start value for the security event, must be in range of 1...65535
	Variation	–
	contextData	
	Type	IdsM _{EventName}_ContextDataType
	Direction	IN
	Comment	Pointer to optional context data. Use NULL_PTR if no context data is available.
	Variation	EventName = {ecuc(IdsM/IdsMConfiguration/IdsMEvent.SHORT-NAME)}
	contextDataSize	
	Type	uint16
	Direction	IN
	Comment	Size of context data, must be in range of 0...<Size of maximum configured context data buffer>.
	Variation	–
Possible Errors	–	

⌋()

8.7.4 IdsM_CustomTimestamp

[SWS_IdsM_91029] ⌈

Name	IdsM_CustomTimestamp		
Comment	Interface to request custom timestamps from the application.		
IsService	true		
Variation	{ecuc(IdsM/IdsMGeneral/IdsMTimeStampOption)} == Custom		
Possible Errors	0	E_OK	Operation successful
	1	E_NOT_OK	Operation failed

Operation	Get	
Comment	This function shall request custom timestamps from the application.	
Variation	–	
Parameters	timestamp	
	Type	uint64
	Direction	OUT
	Comment	Timestamp requested by the IdsM from a custom time source.
	Variation	–
Possible Errors	E_OK E_NOT_OK	

⌋()

8.7.5 Implementation Data Types

8.7.6 IdsM_ContextDataType

[SWS_IdsM_91016] ⌈

Name	IdsM _{EventName}_ContextDataType		
Kind	Type		
Derived from	uint8		
Range	0..255	–	–
Description	Data type for IdsM context data		
Variation	EventName = {ecuc(IdsM/IdsMConfiguration/IdsMEvent.SHORT-NAME)}		
Available via	IdsM_Types.h		

⌋()

8.7.7 IdsM_SecurityEventIdType

[SWS_IdsM_91031] ⌈

Name	IdsM_SecurityEventIdType		
Kind	Type		
Derived from	uint16		
Range	0..65535	–	–
Description	Data type used for local IdsM Security Event IDs		
Variation	–		
Available via	IdsM_Types.h		

⌋()

8.7.8 Ports

8.7.8.1 Port IdsM_IdsMService

[SWS_IdsM_91030] ⌈

Name	IdsMService_{EventName}		
Kind	ProvidedPort	***Interface***	IdsMService_{EventName}
Description	–		
Port Defined Argument Value(s)	***Type***	IdsM_SecurityEventIdType	
	Value	–	
Variation	({ecuc(IdsM/IdsMConfiguration/IdsMEvent/IdsMExternalEventId)} is in range 0x8000 - 0x FFFE)({ecuc(IdsM/IdsMConfiguration/IdsMEvent/IdsMServiceInterfaceOptions)} EXISTS)({ecuc(IdsM/IdsMConfiguration/IdsMEvent/IdsMServiceInterfaceOptions/IdsMAdditional ParameterOption)} = None) EventName = {ecuc(IdsM/IdsMConfiguration/IdsMEvent.SHORT-NAME)}		

⌋()

8.7.8.2 Port IdsM_IdsMSmartSensorService

[SWS_IdsM_91025] ⌈

Name	IdsMSmartSensorService_{EventName}		
Kind	ProvidedPort	***Interface***	IdsMSmartSensorService_{EventName}
Description	–		
Port Defined Argument Value(s)	***Type***	IdsM_SecurityEventIdType	
	Value	–	
Variation	({ecuc(IdsM/IdsMConfiguration/IdsMEvent/IdsMExternalEventId)} is in range 0x8000 - 0x FFFE)({ecuc(IdsM/IdsMConfiguration/IdsMEvent/IdsMServiceInterfaceOptions)} EXISTS)({ecuc(IdsM/IdsMConfiguration/IdsMEvent/IdsMServiceInterfaceOptions/IdsMAdditional ParameterOption)} != None) EventName = {ecuc(IdsM/IdsMConfiguration/IdsMEvent.SHORT-NAME)}		

⌋()

8.7.8.3 Port IdsM_CustomTimestamp

[SWS_IdsM_91026] ⌈

Name	IdsM_CustomTimestamp		
Kind	RequiredPort	***Interface***	IdsM_CustomTimestamp
Description	–		

▽

△

Variation	–

⌋()

9 Sequence diagrams

9.1 Proposal for DEM / Sem Sequence Diagram

Figure 9.1 shows the sequence diagram for the interaction of the IdsM with the *Dem sink*.

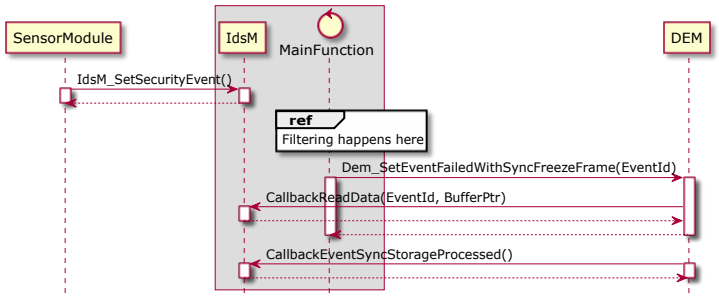

Figure 9.1: Dem sink for Single / Multipartition use case

9.2 Timestamp Sequence Diagrams

Figure 9.2 shows the sequence diagram for the interaction of the IdsM with the *StbM* as timestamp source for the timestamp with AUTOSAR format.

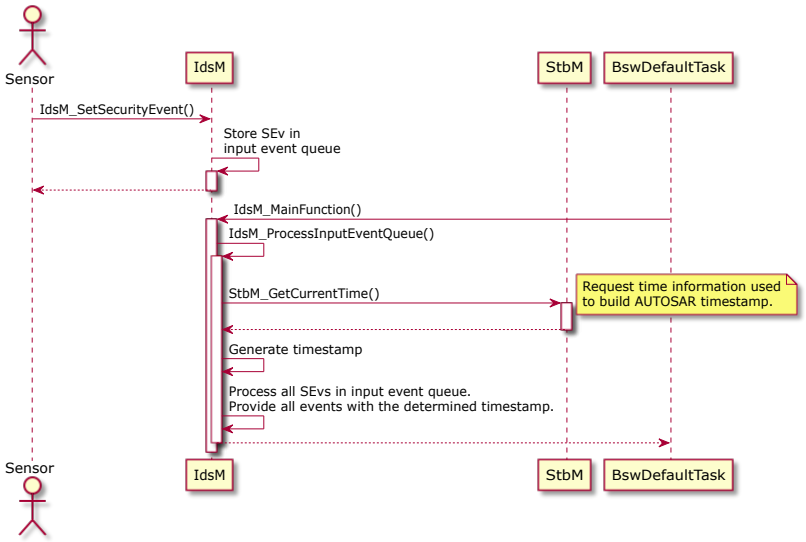

Figure 9.2: AUTOSAR Timestamp: The StbM is used as source for timestamp data

Figure 9.3 shows the sequence diagram for the interaction of the IdsM with the *SW-C* as timestamp source for the timestamp with custom format.

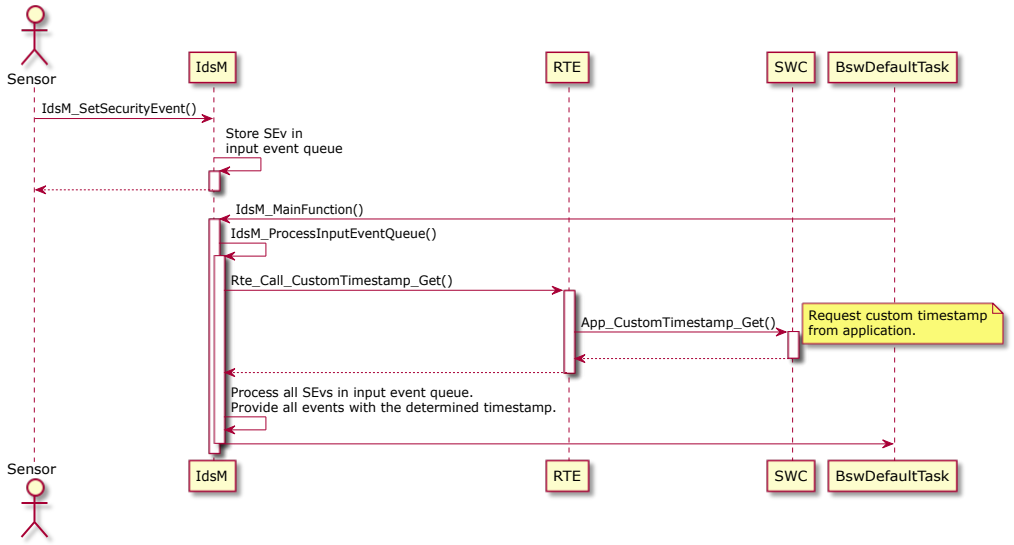

Figure 9.3: Custom Timestamp: Timestamps are requested from the application

10 Configuration specification

In general, this chapter defines configuration parameters and their clustering into containers.

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

Chapter 10.2 lists constraints on the configuration of the IdsM.

Chapter 10.3 specifies published information of the module IdsM.

10.1 Containers and configuration parameters

The following chapters summarize all configuration parameters. The detailed meanings of the parameters described in chapter 7 and chapter 8.

10.1.1 IdsM

Module SWS Item	ECUC_IdsM_00001	
Module Name	IdsM	
Module Description		
Post-Build Variant Support	false	
Supported Config Variants	VARIANT-PRE-COMPILE	
Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
IdsMConfiguration	1	Configuration parameters of the module IdsM. **Tags:** atp.Status=draft
IdsMGeneral	1	General configuration parameters of IdsM. **Tags:** atp.Status=draft

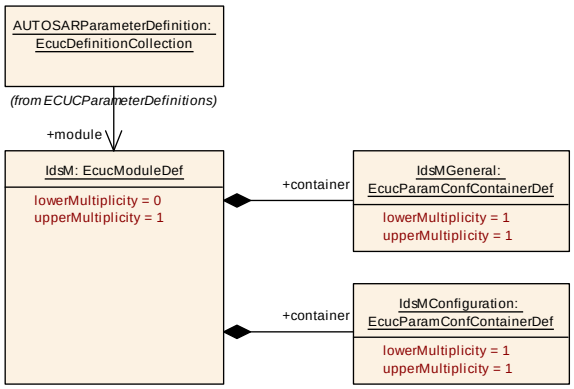

Figure 10.1: Intrusion Detection System Manager Overview

10.1.2 IdsMGeneral

SWS Item	[ECUC_IdsM_00002]
Container Name	IdsMGeneral
Parent Container	IdsM
Description	General configuration parameters of IdsM. **Tags:** atp.Status=draft
Configuration Parameters	

Name	IdsMDevErrorDetect [ECUC_IdsM_00005]		
Parent Container	IdsMGeneral		
Description	This parameter enables/disables the Development Error Detection and Notification. true: Development error detection is enabled. false: Development error detection is disabled. Note: In general, the development error detection is recommended during pre-test phase. It is not recommended to enable the development error detection in production code due to increased runtime and ROM needs. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	IdsMDiagnosticSupport [ECUC_IdsM_00010]		
Parent Container	IdsMGeneral		
Description	Enables or disables the Dcm APIs which are used to read and write certain values of the IdsM module through the diagnostic communication manager. true: Dcm APIs are enabled false: Dcm APIs are disabled **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value	false		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	IdsMInstanceId [ECUC_IdsM_00007]		
Parent Container	IdsMGeneral		
Description	The unique identifier of the sending IdsM instance. This ID helps identifying the origin of a SEv, together with the SEv configuration parameters: ExternalEventId and the IdsMSensorInstanceId. Note: There is only one IdsM (from the AUTOSAR Classic Platform) instance per ECU. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 1023		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	IdsMMainFunctionPeriod [ECUC_IdsM_00004]		
Parent Container	IdsMGeneral		
Description	The period between successive calls to the IdsM main function (as float in seconds). **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	[0 .. INF]		
Default Value	0.01		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	IdsMSignatureGenerateResultLength [ECUC_IdsM_00011]		
Parent Container	IdsMGeneral		
Description	This parameter defines the length in bytes of the signature calculated by the crypto services. This parameter should be set when the IdsMSignatureSupport is enabled. **Tags:** atp.Status=draft		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 4294967295		
Default Value			
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	IdsMSignatureSupport [ECUC_IdsM_00009]		
Parent Container	IdsMGeneral		
Description	This parameter enables/disables the functionality of sending messages to the network with a signature of encryption calculated by the crypto services. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value	false		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	IdsMTimestampOption [ECUC_IdsM_00012]		
Parent Container	IdsMGeneral		
Description	This parameter enables/disables the functionality of having a timestamp field as part of a QSEv and if the origin of the timestamp is from the AUTOSAR stack or from the application (custom timestamp). **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	AUTOSAR		**Tags:** atp.Status=draft
	Custom		**Tags:** atp.Status=draft
	None		**Tags:** atp.Status=draft
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	IdsMVersionInfoApi [ECUC_IdsM_00006]		
Parent Container	IdsMGeneral		
Description	This parameter enables/disables the function IdsM_GetVersionInfo() to get major, minor and patch version information of the module. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	IdsMCsmJobReference [ECUC_IdsM_00015]		
Parent Container	IdsMGeneral		
Description	This parameter references the Csm job that is used to generate signatures when qualified security events must be signed. **Tags:** atp.Status=draft		
Multiplicity	0..1		
Type	Reference to CsmJob		
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		

Name	IdsMNvmBlockDescriptor [ECUC_IdsM_00013]		
Parent Container	IdsMGeneral		
Description	Choose a NvM block descriptor reference, that is used to load and store the non-volatile data of IdsM module. The supported NvM block names are: RAM: IdsM_NvMRamBlockData ROM: IdsM_NvMRomBlockData **Tags:** atp.Status=draft		
Multiplicity	0..1		
Type	Reference to NvMBlockDescriptor		
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		

Name	IdsMStbMSynchronizedTimeBaseReference [ECUC_IdsM_00014]		
Parent Container	IdsMGeneral		
Description	This parameter references the time source when the origin of the timestamp is AUTOSAR. **Tags:** atp.Status=draft		
Multiplicity	0..1		
Type	Reference to StbMSynchronizedTimeBase		
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		

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
IdsMGlobalRate LimitationFilters	0..1	Global rate limitation filters for all SEvs. **Tags:** atp.Status=draft

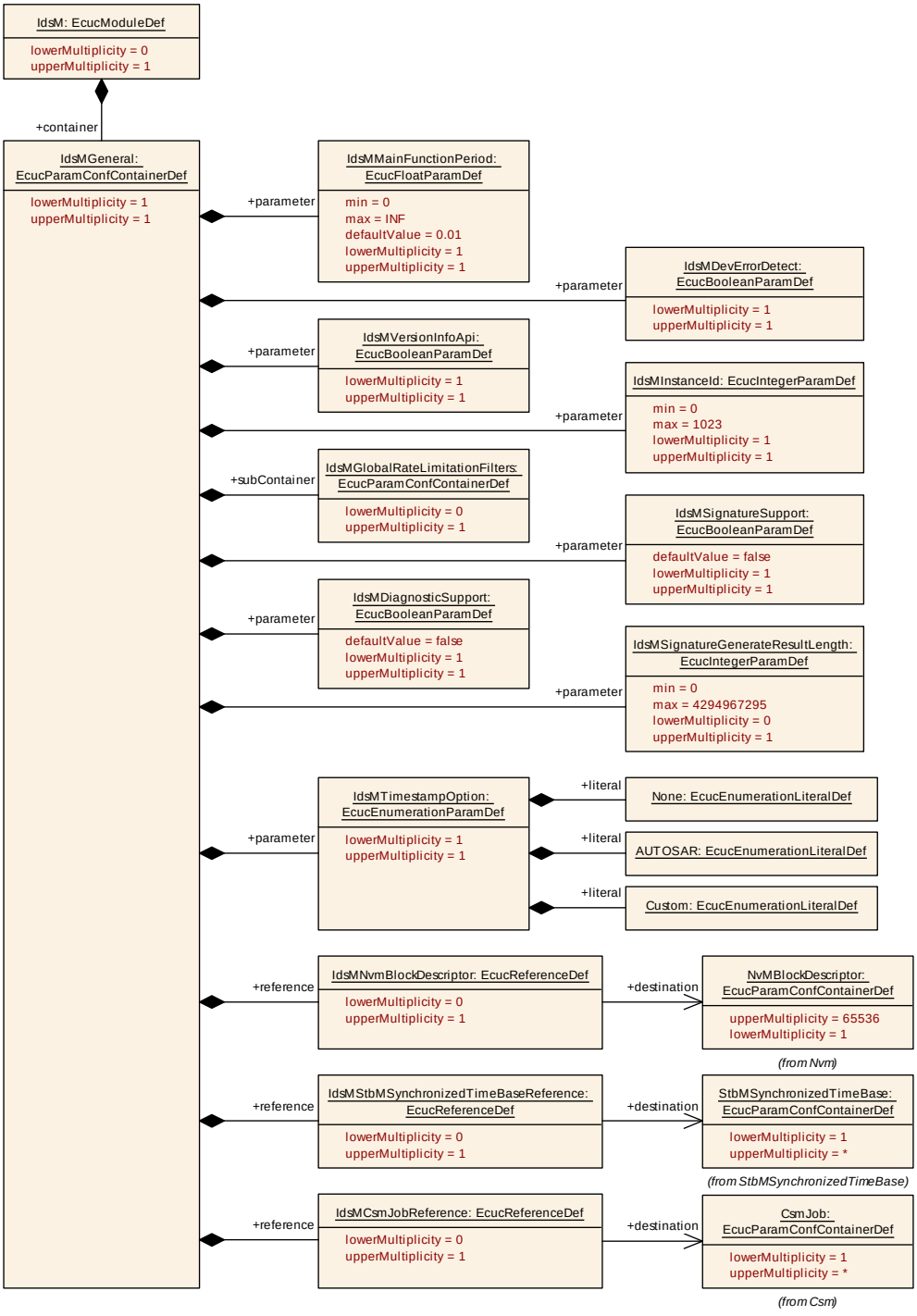

Figure 10.2: IdsM general configuration overview

10.1.3 IdsMGlobalRateLimitationFilter

SWS Item	[ECUC_IdsM_00008]		
Container Name	IdsMGlobalRateLimitationFilters		
Parent Container	IdsMGeneral		
Description	Global rate limitation filters for all SEvs. **Tags:** atp.Status=draft		
Post-Build Variant Multiplicity	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
IdsMFilterEventRate Limitation	0..1	For configurable time intervals of length "IdsMRateLimitationTimeInterval" this filter forwards all the SEvs until reaching the limit "IdsMRateLimitationMaximumEvents". The limit is measured in number of incoming SEvs. Until the end of the time interval, all subsequent SEvs are dropped. This is helpful to cap the load that the IdsM generates unto information sinks like the IdsR. This filter is not specific to a single SEv but it applies to all SEvs handled by the current IdsM instance. Note: Each possible SEv counts as a single one, regardless of its counter value. **Tags:** atp.Status=draft

IdsMFilterTrafficLimitation	0..1	The traffic limitation filter forwards all the incoming SEvs until reaching the limit "IdsMTrafficLimitationMaximumBytes". The limit is measured in incoming amount of bytes. This filter forwards SEvs only, if the accumulated sizes of all incoming SEvs in the current traffic limitation time interval up until the current SEv is smaller or equal than a configurable maximum number of bytes "IdsMTrafficLimitationMaximumBytes". The length of the traffic limitation time interval is configurable in "IdsMTrafficLimitationTimeInterval". This filter is not specific to a single SEv but it applies to all SEvs handled by the current IdsM instance. **Tags:** atp.Status=draft

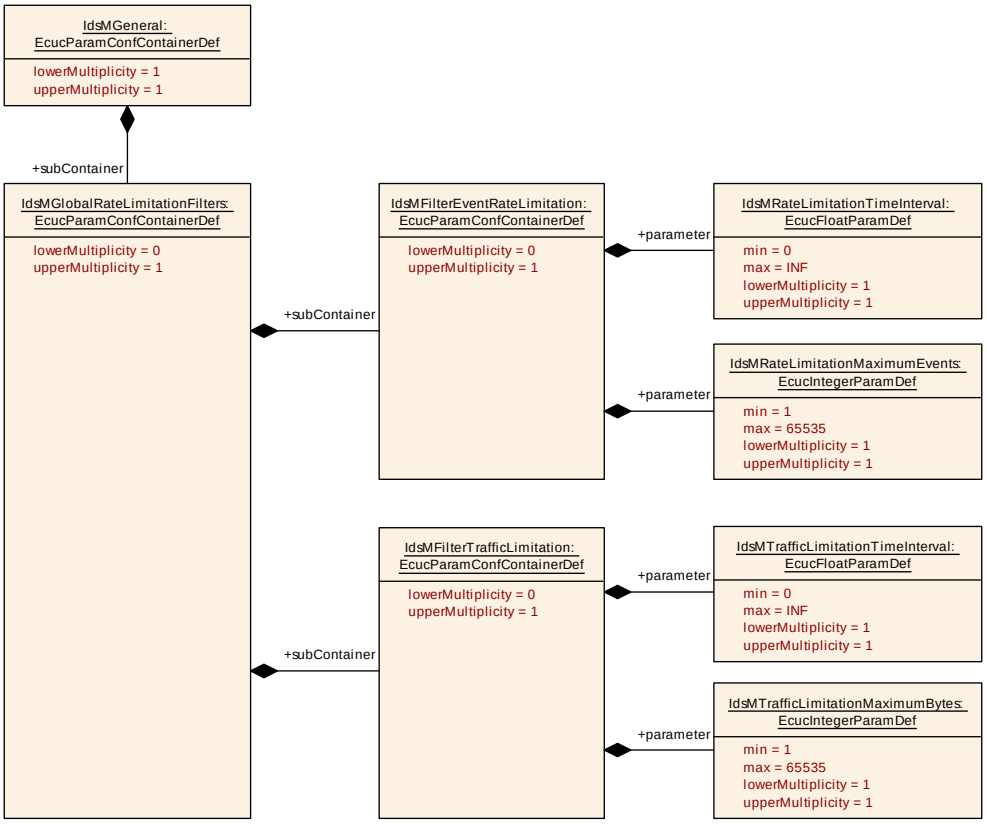

Figure 10.3: IdsM global rate limitation overview

10.1.4 IdsMFilterEventRateLimitation

SWS Item	[ECUC_IdsM_00053]

Container Name	IdsMFilterEventRateLimitation		
Parent Container	IdsMGlobalRateLimitationFilters		
Description	For configurable time intervals of length "IdsMRateLimitationTimeInterval" this filter forwards all the SEvs until reaching the limit "IdsMRateLimitationMaximumEvents". The limit is measured in number of incoming SEvs. Until the end of the time interval, all subsequent SEvs are dropped. This is helpful to cap the load that the IdsM generates unto information sinks like the IdsR. This filter is not specific to a single SEv but it applies to all SEvs handled by the current IdsM instance. Note: Each possible SEv counts as a single one, regardless of its counter value. **Tags:** atp.Status=draft		
Post-Build Variant Multiplicity	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

Name	IdsMRateLimitationMaximumEvents [ECUC_IdsM_00055]		
Parent Container	IdsMFilterEventRateLimitation		
Description	The maximum number of SEvs which are passed on by this filter in a single rate limitation time interval. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 65535		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	IdsMRateLimitationTimeInterval [ECUC_IdsM_00054]		
Parent Container	IdsMFilterEventRateLimitation		
Description	Time interval length of the event rate limitation filter (as float in seconds). Note: Shall be configured as a multiple of the IdsM main function period. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	[0 .. INF]		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

No Included Containers

10.1.5 IdsMFilterTrafficLimitation

SWS Item	[ECUC_IdsM_00056]
Container Name	IdsMFilterTrafficLimitation
Parent Container	IdsMGlobalRateLimitationFilters
Description	The traffic limitation filter forwards all the incoming SEvs until reaching the limit "IdsMTrafficLimitationMaximumBytes". The limit is measured in incoming amount of bytes. This filter forwards SEvs only, if the accumulated sizes of all incoming SEvs in the current traffic limitation time interval up until the current SEv is smaller or equal than a configurable maximum number of bytes "IdsMTrafficLimitationMaximumBytes". The length of the traffic limitation time interval is configurable in "IdsMTrafficLimitationTimeInterval". This filter is not specific to a single SEv but it applies to all SEvs handled by the current IdsM instance. **Tags:** atp.Status=draft
Post-Build Variant Multiplicity	false

Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

Name	IdsMTrafficLimitationMaximumBytes [ECUC_IdsM_00058]		
Parent Container	IdsMFilterTrafficLimitation		
Description	The maximum number of bytes to be sent out by the IdsM in a single traffic limitation time interval. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 65535		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	IdsMTrafficLimitationTimeInterval [ECUC_IdsM_00057]		
Parent Container	IdsMFilterTrafficLimitation		
Description	Length of the traffic limitation time interval (as float in seconds). Note: Shall be configured as a multiple of the IdsM main function period. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	[0 .. INF]		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

No Included Containers

10.1.6 IdsMConfiguration

SWS Item	[ECUC_IdsM_00003]
Container Name	IdsMConfiguration
Parent Container	IdsM
Description	Configuration parameters of the module IdsM. **Tags:** atp.Status=draft
Configuration Parameters	

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
IdsMBlockState	0..16	Configuration of an IdsM blocking state used in the IdsMStateBlockFilter to suspend the collection of security events. The active state is reported by the BswM via IdsM_BswM_StateChanged(). **Tags:** atp.Status=draft
IdsMBufferConfiguration	1	Configuration of the event buffers and context data buffers used by IdsM. **Tags:** atp.Status=draft
IdsMEvent	1..65535	Configuration of the IdsM Event unit which is reported by a sensor and its parameters. **Tags:** atp.Status=draft
IdsMFilterChain	0..*	A filter chain is a combination of filters that affects one or more SEvs. A filter receives a SEv, checks condition(s) and, e.g. - forwards SEv immediately/later - drops SEv - stores SEv - modifies SEv Consider that the filter order is defined as follows: - Reporting Mode Level (per SEv ID) - Block State (per SEv ID) - Forward Every nth (per SEv ID) - Event Aggregation (per SEv ID) - Event Threshold (per SEv ID) - Event Rate Limitation (per IdsM Instance) - Traffic Limitation (per IdsM Instance) **Tags:** atp.Status=draft
IdsMPdus	0..1	Configuration of the PDU references used to send the events data. **Tags:** atp.Status=draft

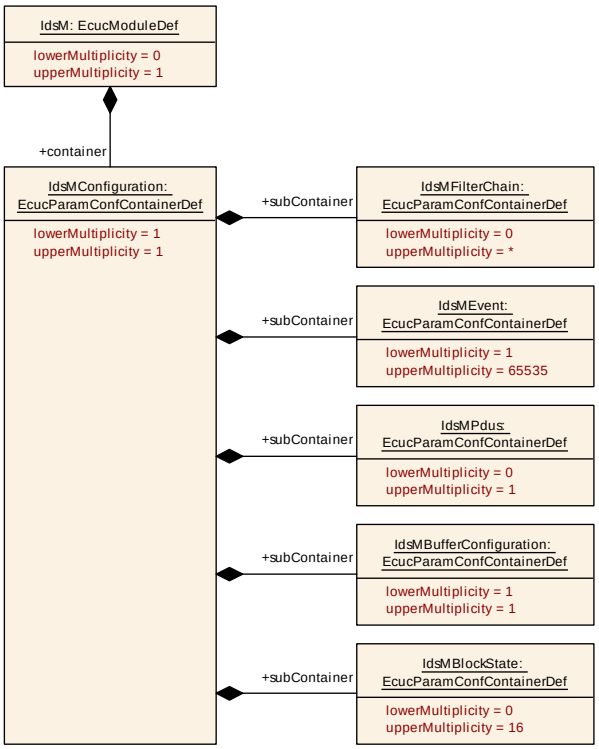

Figure 10.4: IdsM configuration overview

10.1.7 IdsMFilterChain

SWS Item	[ECUC_IdsM_00016]		
Container Name	IdsMFilterChain		
Parent Container	IdsMConfiguration		
Description	A filter chain is a combination of filters that affects one or more SEvs. A filter receives a SEv, checks condition(s) and, e.g. - forwards SEv immediately/later - drops SEv - stores SEv - modifies SEv Consider that the filter order is defined as follows: - Reporting Mode Level (per SEv ID) - Block State (per SEv ID) - Forward Every nth (per SEv ID) - Event Aggregation (per SEv ID) - Event Threshold (per SEv ID) - Event Rate Limitation (per IdsM Instance) - Traffic Limitation (per IdsM Instance) **Tags:** atp.Status=draft		
Post-Build Variant Multiplicity	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
IdsMBlockStateFilter	0..1	This state filter drops SEvs if the current State reported by the BswM is in this state filter list. **Tags:** atp.Status=draft
IdsMEventAggregation Filter	0..1	All received events of a certain event ID that are received by this filter during a single aggregation time interval are not forwarded immediately. Instead, only the last or the first received SEv is stored in an aggregation buffer, depending on the configuration of "IdsMContextDataSourceSelector". The counter field of the SEv is modified so that it contains the sum of the counter fields of all incoming SEvs during the current aggregation time interval. At the end of the aggregation time interval, the buffered SEv is sent out and the aggregation buffer is cleared. If there was no incoming SEv until the end of the aggregation time interval, no message will be sent. **Tags:** atp.Status=draft
IdsMEventThresholdFilter	0..1	During each time interval "IdsMEventThresholdTimeInterval", the filter drops the first "IdsMEventThresholdNumber - 1" SEvs and forwards all other incoming SEvs immediately until the end of the time interval. **Tags:** atp.Status=draft
IdsMForwardEveryNth Filter	0..1	Out of all incoming SEVs, drop all but every nth. Those will be forwarded without modification. **Tags:** atp.Status=draft

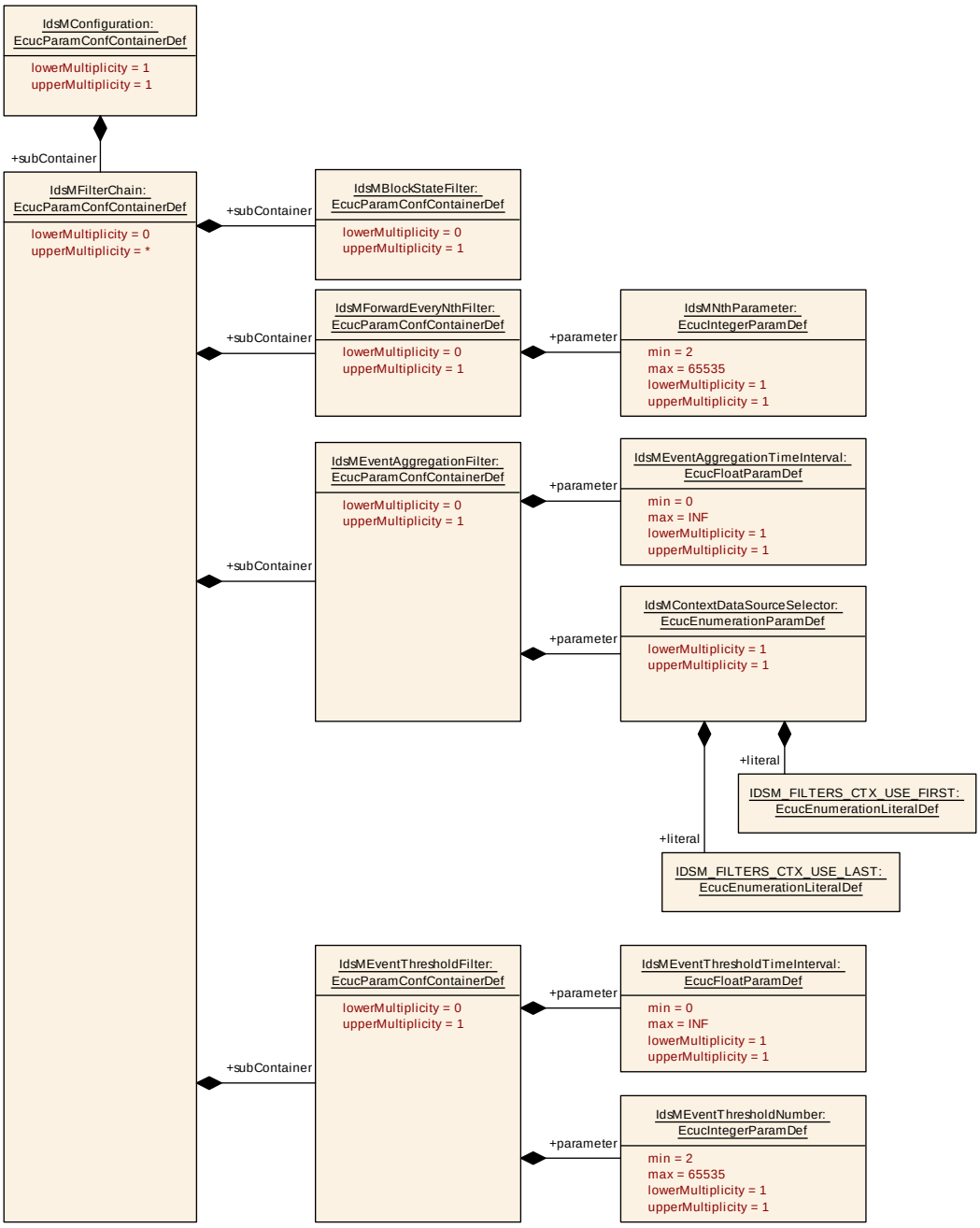

Figure 10.5: IdsM filter chain overview

10.1.8 IdsMBlockStateFilter

SWS Item	[ECUC_IdsM_00021]
Container Name	IdsMBlockStateFilter
Parent Container	IdsMFilterChain

Description	This state filter drops SEvs if the current State reported by the BswM is in this state filter list. **Tags:** atp.Status=draft		
Post-Build Variant Multiplicity	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

Name	IdsMBlockStateReference [ECUC_IdsM_00051]		
Parent Container	IdsMBlockStateFilter		
Description	The collection of SEvs during this state will be suspended. **Tags:** atp.Status=draft		
Multiplicity	1..16		
Type	Reference to IdsMBlockState		
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

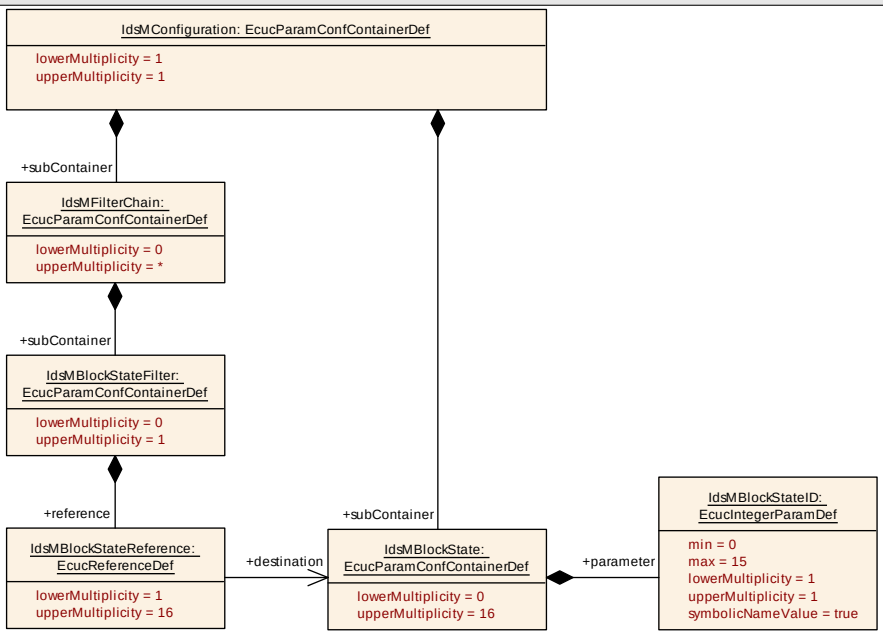

Figure 10.6: IdsM block state filter overview

10.1.9 IdsMBlockState

SWS Item	[ECUC_IdsM_00020]		
Container Name	IdsMBlockState		
Parent Container	IdsMConfiguration		
Description	Configuration of an IdsM blocking state used in the IdsMStateBlockFilter to suspend the collection of security events. The active state is reported by the BswM via IdsM_BswM_StateChanged(). **Tags:** atp.Status=draft		
Post-Build Variant Multiplicity	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

Name	IdsMBlockStateID [ECUC_IdsM_00052]		
Parent Container	IdsMBlockState		
Description	This value specifies the identifier of this block state. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 15		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

No Included Containers

10.1.10 IdsMForwardEveryNthFilter

SWS Item	[ECUC_IdsM_00022]		
Container Name	IdsMForwardEveryNthFilter		
Parent Container	IdsMFilterChain		
Description	Out of all incoming SEVs, drop all but every nth. Those will be forwarded without modification. **Tags:** atp.Status=draft		
Post-Build Variant Multiplicity	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

Name	IdsMNthParameter [ECUC_IdsM_00023]	
Parent Container	IdsMForwardEveryNthFilter	
Description	For each SEv ID for which this filter is configured, this parameter assigns the appropriate n. Only 1 from n SEvs will be forwarded. **Tags:** atp.Status=draft	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	2 .. 65535	

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

No Included Containers

10.1.11 IdsMEventAggregationFilter

SWS Item	[ECUC_IdsM_00024]		
Container Name	IdsMEventAggregationFilter		
Parent Container	IdsMFilterChain		
Description	All received events of a certain event ID that are received by this filter during a single aggregation time interval are not forwarded immediately. Instead, only the last or the first received SEv is stored in an aggregation buffer, depending on the configuration of "IdsMContextDataSourceSelector". The counter field of the SEv is modified so that it contains the sum of the counter fields of all incoming SEvs during the current aggregation time interval. At the end of the aggregation time interval, the buffered SEv is sent out and the aggregation buffer is cleared. If there was no incoming SEv until the end of the aggregation time interval, no message will be sent. **Tags:** atp.Status=draft		
Post-Build Variant Multiplicity	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

Name	IdsMContextDataSourceSelector [ECUC_IdsM_00026]		
Parent Container	IdsMEventAggregationFilter		
Description	The resulting SEv from the aggregation filter contains the context data from one of the following two sources: IDSM_FILTERS_CTX_USE_FIRST = ContextData of first received SEv is used for resulting QSEv. IDSM_FILTERS_CTX_USE_LAST = ContextData of last received SEv is used for resulting QSEv. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	IDSM_FILTERS_CTX_USE_FIRST	**Tags:** atp.Status=draft	
	IDSM_FILTERS_CTX_USE_LAST	**Tags:** atp.Status=draft	
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	IdsMEventAggregationTimeInterval [ECUC_IdsM_00025]		
Parent Container	IdsMEventAggregationFilter		
Description	Length of the aggregation time interval (as float in seconds). Note: Shall be configured as a multiple of the IdsM main function period. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	[0 .. INF]		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

No Included Containers

10.1.12 IdsMEventThresholdFilter

SWS Item	[ECUC_IdsM_00027]		
Container Name	IdsMEventThresholdFilter		
Parent Container	IdsMFilterChain		
Description	During each time interval "IdsMEventThresholdTimeInterval", the filter drops the first "IdsMEventThresholdNumber - 1" SEvs and forwards all other incoming SEvs immediately until the end of the time interval. **Tags:** atp.Status=draft		
Post-Build Variant Multiplicity	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

Name	IdsMEventThresholdNumber [ECUC_IdsM_00029]		
Parent Container	IdsMEventThresholdFilter		
Description	This parameter assigns the threshold ’p’ for each SEv ID affected by this threshold filter. All SEvs ’ p-1’ are dropped, SEvs equal or greater than ’p’ are forwarded. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	2 .. 65535		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	IdsMEventThresholdTimeInterval [ECUC_IdsM_00028]		
Parent Container	IdsMEventThresholdFilter		
Description	Length of the threshold time interval (as float in seconds). Note: Shall be configured as a multiple of the IdsM main function period. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	[0 .. INF]		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

No Included Containers

10.1.13 IdsMEvent

SWS Item	[ECUC_IdsM_00017]		
Container Name	IdsMEvent		
Parent Container	IdsMConfiguration		
Description	Configuration of the IdsM Event unit which is reported by a sensor and its parameters. **Tags:** atp.Status=draft		
Post-Build Variant Multiplicity	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

Name	IdsMExternalEventId [ECUC_IdsM_00032]		
Parent Container	IdsMEvent		
Description	The external security event ID which is reported to the sink. There are two different value ranges depending on the referencing module: Standarized SEv ID is defined by the AUTOSAR specification. This ID is usually derived from the SecXT. Standard ID range: 0x0000 - 0x8000 Generic User Event ID is defined by the user. Used when a SW-C / Application references the SEv. Generic ID range: 0x8000 - 0xFFFE. 0xFFFF is considered an invalid ID **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 65534		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	IdsMInternalEventId [ECUC_IdsM_00033]		
Parent Container	IdsMEvent		
Description	Consecutive number used internally as an identifier by the IdsM module. This number is calculated internally and shall not be configured manually. This parameter is only available to publish the result of this calculation. Applications using IdsM APIs shall not rely on the value of this parameter. Instead, they shall use the symbolic name value. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 65535		
Default Value	65535		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	IdsMReportingModeFilter [ECUC_IdsM_00036]		
Parent Container	IdsMEvent		
Description	The reporting mode filter defines the level of detail of the reporting. Whether SEv should be dropped, forwarded with context data or forwarded without context data. The parameter determines if the SEv is either: - dropped (OFF) - sent without context data (BRIEF) - sent with context data (DETAILED) - sent without context data, ignoring the rest of the filter chain (BRIEF_BYPASSING_FILTERS) - sent with context data ignoring the rest of the filter chain (DETAILED_BYPASSING_FILTERS) **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	BRIEF	**Tags:** atp.Status=draft	
	BRIEF_BYPASSING_FILTERS	**Tags:** atp.Status=draft	
	DETAILED	**Tags:** atp.Status=draft	
	DETAILED_BYPASSING_FILTERS	**Tags:** atp.Status=draft	
	OFF	**Tags:** atp.Status=draft	
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	IdsMSensorInstanceId [ECUC_IdsM_00031]		
Parent Container	IdsMEvent		
Description	The instance ID of the sensor which reports security events to the IdsM. If there is only one instance of a sensor, the default ID is 0. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 65535		
Default Value	0		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	IdsMSinkDem [ECUC_IdsM_00035]		
Parent Container	IdsMEvent		
Description	The QSEv will be sent to the Dem Module into a Security Event Memory (Sem) to persist it on the local ECU. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value	false		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	IdsMSinkIdsR [ECUC_IdsM_00034]		
Parent Container	IdsMEvent		
Description	The QSEv will be sent to the IDS Reporter. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value	false		
Post-Build Variant Value	false		

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

Name	IdsMFilterChainRef [ECUC_IdsM_00030]		
Parent Container	IdsMEvent		
Description	Reference to a configured IdsM filter chain. **Tags:** atp.Status=draft		
Multiplicity	0..1		
Type	Reference to IdsMFilterChain		
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		

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
IdsMServiceInterface Options	0..1	Adittional configuration parameters of a SEv when the sensor is a SW-C or application. **Tags:** atp.Status=draft

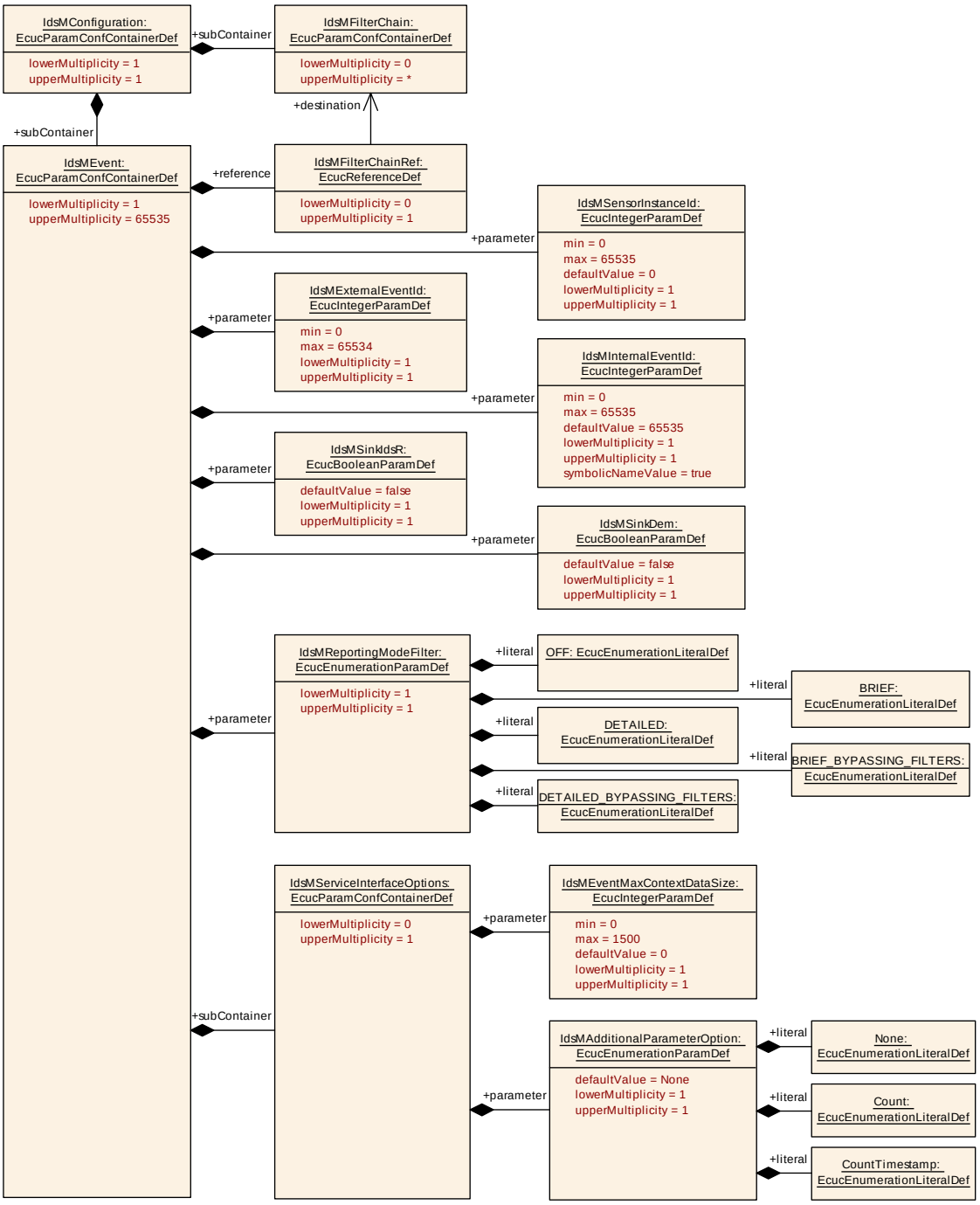

Figure 10.7: IdsM event overview

10.1.14 IdsMReportingModeFilter

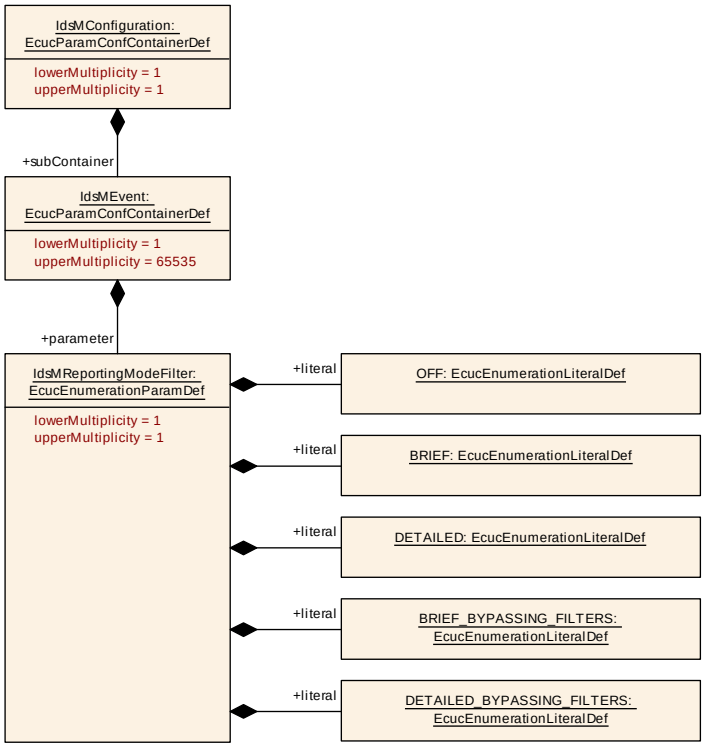

Figure 10.8: IdsM reporting mode filter overview

10.1.15 IdsMPdus

SWS Item	[ECUC_IdsM_00018]		
Container Name	IdsMPdus		
Parent Container	IdsMConfiguration		
Description	Configuration of the PDU references used to send the events data. **Tags:** atp.Status=draft		
Post-Build Variant Multiplicity	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
IdsMIfTxPdu	0..1	IF PDU used to transmit a QSEv via the PduR to the IdsR. If the total size of the QSEv's data to be transmitted fits in a single frame of the underlying bus, the IF PDU is used. **Tags:** atp.Status=draft
IdsMTpTxPdu	0..1	TP PDU used to transmit a QSEv via the PduR to the IdsR. If the total size of the QSEv's data to be transmitted is bigger than the size of a single frame of the underlying bus, the TP PDU is used. **Tags:** atp.Status=draft

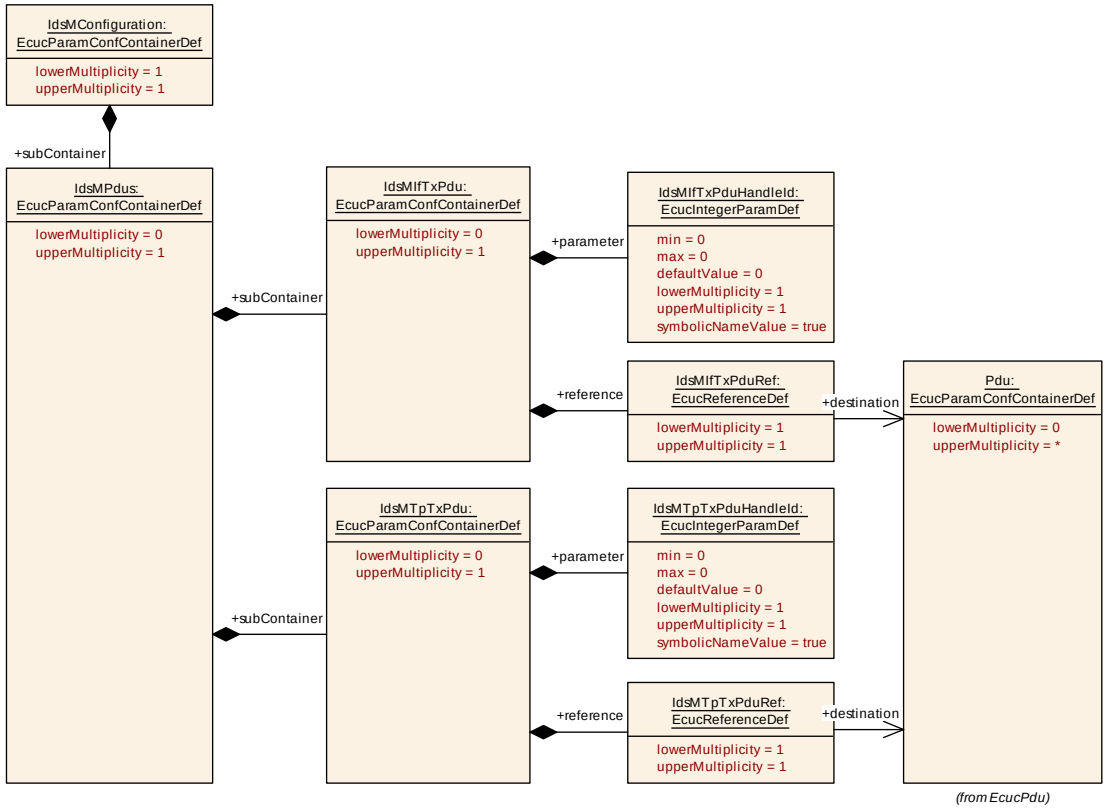

Figure 10.9: IdsM Pdus overview

10.1.16 IdsMIfTxPdu

SWS Item	[ECUC_IdsM_00040]

Container Name	IdsMIfTxPdu		
Parent Container	IdsMPdus		
Description	IF PDU used to transmit a QSEv via the PduR to the IdsR. If the total size of the QSEv's data to be transmitted fits in a single frame of the underlying bus, the IF PDU is used. **Tags:** atp.Status=draft		
Post-Build Variant Multiplicity	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

Name	IdsMIfTxPduHandleId [ECUC_IdsM_00041]		
Parent Container	IdsMIfTxPdu		
Description	IdsM does not use this parameter, content will be ignored. The existence of this parameter is needed by PduR. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 0		
Default Value	0		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	IdsMIfTxPduRef [ECUC_IdsM_00042]
Parent Container	IdsMIfTxPdu
Description	Reference to the IF PDU used for transmission of the QSEvs. **Tags:** atp.Status=draft
Multiplicity	1
Type	Reference to Pdu
Post-Build Variant Value	false

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

No Included Containers

10.1.17 IdsMEventTpTxPdu

SWS Item	[ECUC_IdsM_00043]		
Container Name	IdsMTpTxPdu		
Parent Container	IdsMPdus		
Description	TP PDU used to transmit a QSEv via the PduR to the IdsR. If the total size of the QSEv's data to be transmitted is bigger than the size of a single frame of the underlying bus, the TP PDU is used. **Tags:** atp.Status=draft		
Post-Build Variant Multiplicity	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

Name	IdsMTpTxPduHandleId [ECUC_IdsM_00044]		
Parent Container	IdsMTpTxPdu		
Description	IdsM does not use this parameter, content will be ignored. The existence of this parameter is needed by PduR. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 0		
Default Value	0		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	IdsMTpTxPduRef [ECUC_IdsM_00045]		
Parent Container	IdsMTpTxPdu		
Description	Reference to the TP PDU used for transmission of the QSEvs. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	Reference to Pdu		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

No Included Containers

10.1.18 IdsMBufferConfiguration

SWS Item	[ECUC_IdsM_00019]
Container Name	IdsMBufferConfiguration
Parent Container	IdsMConfiguration
Description	Configuration of the event buffers and context data buffers used by IdsM. **Tags:** atp.Status=draft
Configuration Parameters	

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
IdsMContextDataBuffer	0..65535	Buffer that is reserved to store the context data of SEvs. Depending on the type of SEv that is processed, there can be significant differences in sizes of the context data. **Tags:** atp.Status=draft
IdsMEventBuffers	1	Buffers used to store the SEvs. **Tags:** atp.Status=draft

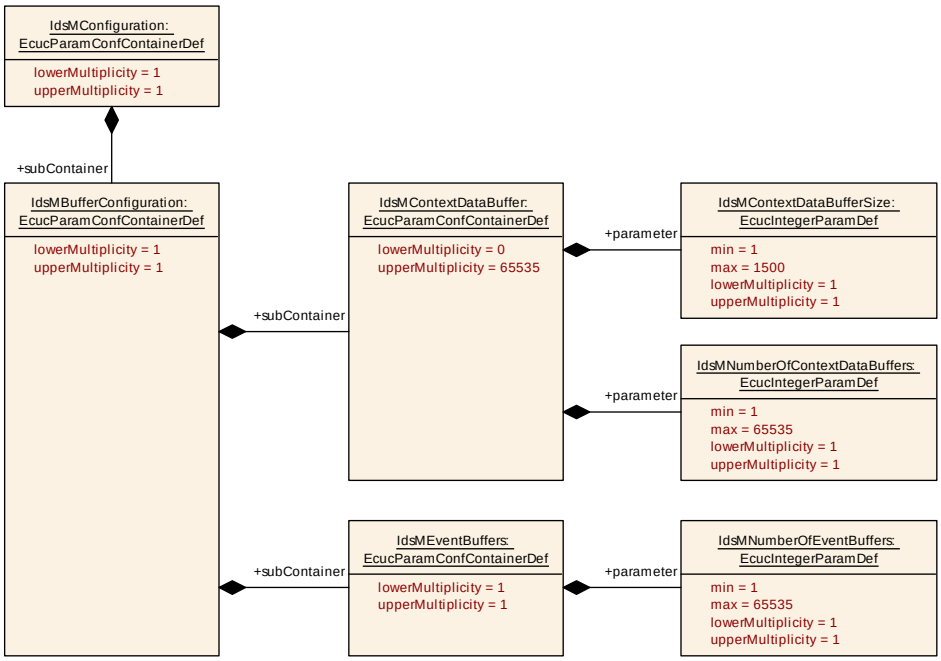

Figure 10.10: IdsM buffer configuration overview

10.1.19 IdsMContextDataBuffer

SWS Item	[ECUC_IdsM_00046]		
Container Name	IdsMContextDataBuffer		
Parent Container	IdsMBufferConfiguration		
Description	Buffer that is reserved to store the context data of SEvs. Depending on the type of SEv that is processed, there can be significant differences in sizes of the context data. **Tags:** atp.Status=draft		
Post-Build Variant Multiplicity	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

Name	IdsMContextDataBufferSize [ECUC_IdsM_00047]		
Parent Container	IdsMContextDataBuffer		
Description	Size of the context data buffer in bytes. It is recommended to configure buffers with an appropriate size depending on the configured SEvs. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 1500		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	IdsMNumberOfContextDataBuffers [ECUC_IdsM_00048]		
Parent Container	IdsMContextDataBuffer		
Description	The number of buffers with the configured buffer size specified in IdsMContextDataBufferSize. It is recommended to configure an appropriate number of buffers depending on the configured SEvs. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 65535		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

No Included Containers

10.1.20 IdsMEventBuffers

SWS Item	[ECUC_IdsM_00049]
Container Name	IdsMEventBuffers
Parent Container	IdsMBufferConfiguration

Description	Buffers used to store the SEvs. **Tags:** atp.Status=draft
Configuration Parameters	

Name	IdsMNumberOfEventBuffers [ECUC_IdsM_00050]		
Parent Container	IdsMEventBuffers		
Description	The number of event buffers used to store the SEvs. The suggested number of buffers can be calculated as follows: IdsMNumberOfBuffers = Number of Aggregation Filter Instances + Upper bound of parallel processed SEvs. Number of Aggregation Filter Instances = The number of configured SEvs that use a filter chain that contains an aggregation filter. Upper bound of parallel processed SEvs = 10% of the number of configured SEvs. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 65535		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

No Included Containers

10.1.21 IdsMServiceInterfaceOptions

SWS Item	[ECUC_IdsM_00037]
Container Name	IdsMServiceInterfaceOptions
Parent Container	IdsMEvent
Description	Adittional configuration parameters of a SEv when the sensor is a SW-C or application. **Tags:** atp.Status=draft

Post-Build Variant Multiplicity	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

Name	IdsMAdditionalParameterOption [ECUC_IdsM_00039]	
Parent Container	IdsMServiceInterfaceOptions	
Description	In addition to the optional context data the Service Port Interface can be extended by the following parameters: - None: No extensions - Count: Additionally a count can be passed. This value is used as an initialization counter for the corresponding SEv. - Count and Timestamp: Additionally to the count a timestamp can be passed. Note: The timestamp option depends on whether IdsMTimeStampOption is configured to enable timestamps. **Tags:** atp.Status=draft	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	Count	**Tags:** atp.Status=draft
	CountTimestamp	**Tags:** atp.Status=draft
	None	**Tags:** atp.Status=draft
Default Value	None	
Post-Build Variant Value	false	
Value Configuration Class	**Pre-compile time**	X All Variants
	Link time	–
	Post-build time	–
Scope / Dependency	scope: local	

Name	IdsMEventMaxContextDataSize [ECUC_IdsM_00038]		
Parent Container	IdsMServiceInterfaceOptions		
Description	Maximum number of bytes used by the IdsM and the RTE when forwarding context data of the corresponding security event. This parameter is only used for SW-C use cases. This is the maximum amount of bytes defined for transmission of the context data. In case this is a Basic Software Module SEv, the configuration of this parameter is not necessary and will be ignored. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 1500		
Default Value	0		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

No Included Containers

10.2 Configuration Constraints

[SWS_IdsM_CONSTR_00002] ⌈This section lists configuration constraints for the IdsM Module. Instances of the container IdsMFilterChain always require to have at least one filter configured (IdsMBlockStateFilter, IdsMForwardEveryNthFilter, IdsMEventAggregationFilter, IdsMEventThresholdFilter).

⌋()

10.3 Published Information

For details refer to the chapter 10.3 “Published Information” in [3, SWS BSW General].