Document Title	Specification of Intrusion Detection System Manager
Document Category	Software specification (SWS)
Document UID	977
Document Status	published
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	R26-11

Document Change History			
Date	**Release**	**Changed by**	**Description**
2026-11-26	R26-11	AUTOSAR Release Management	• Introduced new Security Event IdsM Statistics • Added Post Build Selectable Support • Tracing to Requirements Specification IntrusionDetectionSystem removed
2025-11-27	R25-11	AUTOSAR Release Management	• Introduced Authenticator with MAC and Signature as authentication option • Introduce Transmission Retry Handling in case sink failed • Add BSW ReportingMode API • Remove obsolete elements • Set Status of Chapter 10 ECUC tables to valid
2024-11-27	R24-11	AUTOSAR Release Management	• Introduction of Context Data Version • Introduction of new Reporting API for Security Events • Storage of Security Events remove draft status and add description • Use BswM transistion to start/stop transmission of QSEvs • Introduce Severity of Security Events
2023-11-23	R23-11	AUTOSAR Release Management	• Introduction of Qualified Event Buffer • New Security Event no Qualified Event Buffer available • New Type Definition IdsM ExternalSecurityEventIdType

▽

△

2022-11-24	R22-11	AUTOSAR Release Management	• Security Events - Added additional context data • Added SWS_IdsM_01034
2021-11-25	R21-11	AUTOSAR Release Management	• Added subcontainers for Timestamp and Signature • Additional Internal Security Events signals communication error • clarification of Internal Security Event • Replaced handdrawn sequence charts by generated
2020-11-30	R20-11	AUTOSAR Release Management	• Initial release

Disclaimer

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

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

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

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

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

Table of Contents

1 Introduction and functional overview 9
2 Acronyms and Abbreviations 10
3 Related documentation 13
3.1 Related Specification . . . 13
3.2 Input Documents & Related Standards and Norms . . . 13
4 Constraints and assumptions 14
4.1 Assumptions . . . 14
4.2 Applicability to Car Domains . . . 14
5 Dependencies to other modules 15
5.1 Interfaces to Modules . . . 15
5.1.1 Sensor Modules . . . 15
5.1.2 Error Handling Modules . . . 16
5.1.3 Diagnostic Access . . . 16
5.1.4 Persistence of Reporting Level . . . 16
5.1.5 IdsR Sink . . . 16
5.1.6 Dem / Sem Sink . . . 16
5.1.7 BSW Scheduler . . . 16
5.2 File Structure . . . 17
5.2.1 Code File Structure . . . 17
5.2.2 Header File Structure . . . 17
6 Requirements Tracing 18
7 Functional specification 19
7.1 Overview . . . 19
7.2 Module Handling . . . 20
7.2.1 Initialization . . . 21
7.2.2 Timing Related Functionality . . . 22
7.3 Reception and Buffering of Events . . . 22
7.3.1 Reception of Events . . . 22
7.3.1.1 Smart Sensors . . . 22
7.3.2 Security Event Definition . . . 23
7.3.3 Buffers . . . 24
7.3.4 Context Data Modifier . . . 26
7.3.5 Context Data Endianess . . . 28
7.4 IdsM Internal SEvs . . . 29
7.5 Qualification of SEvs . . . 30
7.6 Filter Chain . . . 32
7.6.1 Blocker Filters . . . 33
7.6.1.1 Reporting Mode Filter . . . 34

7.6.1.2 Block State Filter . . . 35
7.6.2 Sampling Filters . . . 35
7.6.2.1 Forward Every Nth . . . 35
7.6.3 Aggregation Filters . . . 36
7.6.3.1 Event Aggregation Filter . . . 36
7.6.3.2 Event Threshold Filter . . . 38
7.6.4 Rate Limitation Filters . . . 39
7.6.4.1 Event Rate Limitation . . . 40
7.6.4.2 Traffic Limitation . . . 40
7.7 Timestamp . . . 40
7.8 Reporting and Persistence of SEVs . . . 43
7.8.1 Structure Of QSEVs . . . 43
7.8.2 Propagation of QSEvs: IdsR Sink . . . 44
7.8.2.1 Authenticity of QSEvs . . . 45
7.8.2.2 IDS Service Interface Options . . . 46
7.8.2.3 Transmission Protocols . . . 46
7.8.2.4 Transmission Retry Handling . . . 47
7.8.3 Storage of Events: Dem / Sem Sink . . . 48
7.9 Persistence in NvM of Configuration . . . 49
7.10 Reconfiguration of SEvs . . . 50
7.10.1 Reconfiguration via Diagnostics . . . 50
7.10.2 Reconfiguration via Reporting Mode API . . . 51
7.11 Statistics . . . 52
7.11.1 Configuration IdsM Statistics . . . 53
7.11.2 Security Event IdsM Statistics . . . 54
7.12 Error Classification . . . 57
7.12.1 Development Errors . . . 57
7.12.2 Runtime Errors . . . 57
7.12.3 Production Errors . . . 57
7.13 Error Detection and Notification . . . 57
7.13.1 Api Parameter Checking . . . 58
8 API specification 59
8.1 Imported Types . . . 59
8.2 Type Definitions . . . 59
8.2.1 IdsM_ConfigType . . . 59
8.2.2 IdsM_SecurityEventIdType . . . 60
8.2.3 IdsM_SensorInstanceIdType . . . 60
8.2.4 IdsM_Filters_BlockStateType . . . 60
8.2.5 IdsM_Filters_ReportingModeType . . . 61
8.2.6 IdsM_ReportingModeStatusType . . . 61
8.2.7 IdsM_ReturnType . . . 62
8.2.8 IdsM_TransmissionStateType . . . 62
8.3 Function Definitions . . . 62

8.3.1 IdsM_Init . . . 62
8.3.2 IdsM_ReportSecurityEvent . . . 63
8.3.3 IdsM_GetReportingMode . . . 64
8.3.4 IdsM_SetReportingMode . . . 64
8.3.5 IdsM_GetStatusReportingMode . . . 65
8.3.6 Callback Notifications . . . 65
8.3.6.1 IdsM_BswM_StateChanged . . . 65
8.3.6.2 IdsM_TpTxConfirmation . . . 66
8.3.6.3 IdsM_TxConfirmation . . . 66
8.3.6.4 IdsM_CopyTxData . . . 67
8.3.6.5 IdsM_Dcm_GetReportingMode_RequestResults . . . 68
8.3.6.6 IdsM_Dcm_GetReportingMode_Start . . . 69
8.3.6.7 IdsM_Dcm_SetReportingMode_Start . . . 69
8.3.6.8 IdsM_DemReadQSEv . . . 71
8.3.6.9 IdsM_CsmNotification . . . 71
8.3.6.10 IdsM_TransmissionSetState . . . 72
8.4 Scheduled Functions . . . 72
8.4.1 IdsM_MainFunction . . . 72
8.5 Expected Interfaces . . . 72
8.5.1 Mandatory Interfaces . . . 73
8.5.2 Optional Interfaces . . . 73
8.5.3 Configurable Interfaces . . . 73
8.6 Service Interfaces . . . 74
8.6.1 Client-Server Interfaces . . . 74
8.6.1.1 IdsM_IdsMService(EventName) . . . 74
8.6.1.2 IdsM_RequestCustomTimestamp . . . 75
8.6.1.3 IdsM_ReportingModeService . . . 76
8.6.2 Implementation Data Types . . . 78
8.6.3 Ports . . . 78
8.6.3.1 Port IdsM_IdsMService . . . 78
8.6.3.2 Port IdsM_RequestCustomTimestamp . . . 79
8.6.3.3 Port IdsM_ReportingModeService . . . 79
8.6.4 Callout Interfaces . . . 79
8.6.4.1 IdsMContextDataModifierCallout . . . 79

9 Sequence diagrams 81
9.1 Sequence diagram for storage of qualified security events in Dem . . . 81
9.2 Timestamp Sequence Diagrams . . . 82

10 Configuration specification 84
10.1 Containers and configuration parameters . . . 84
10.1.1 IdsM . . . 84
10.1.2 IdsMGeneral . . . 85
10.1.3 IdsMGlobalRateLimitationFilter . . . 89

10.1.4 IdsMSecurityEventRefs . . . 91
10.1.5 IdsMFilterEventRateLimitation . . . 93
10.1.6 IdsMFilterTrafficLimitation . . . 94
10.1.7 IdsMTimestamp . . . 96
10.1.8 IdsMAuthenticator . . . 97
10.1.9 IdsMConfiguration . . . 98
10.1.10 IdsMFilterChain . . . 100
10.1.11 IdsMBlockStateFilter . . . 101
10.1.12 IdsMForwardEveryNthFilter . . . 102
10.1.13 IdsMEventAggregationFilter . . . 103
10.1.14 IdsMEventThresholdFilter . . . 104
10.1.15 IdsMPdus . . . 106
10.1.16 IdsMIfTxPdu . . . 107
10.1.17 IdsMEventTpTxPdu . . . 108
10.1.18 IdsMTransmissionRetry . . . 110
10.1.19 IdsMBufferConfiguration . . . 111
10.1.20 IdsMContextDataBuffer . . . 112
10.1.21 IdsMEventBuffers . . . 114
10.1.22 IdsMQualifiedEventBuffers . . . 114
10.1.23 IdsMEvent . . . 115
10.1.24 IdsMReportingModeFilter . . . 121
10.1.25 IdsMServiceInterfaceOptions . . . 121
10.1.26 IdsMContextDataModifierOptions . . . 122
10.1.27 IdsMBlockState . . . 123
10.1.28 IdsMContextDataModification . . . 124
10.2 Configuration Constraints . . . 127
10.3 Published Information . . . 128
A Change history of AUTOSAR traceable items 129
A.1 Traceable item history of this document according to AUTOSAR Release R26-11 . . . 129
A.1.1 Added Specification Items in R26-11 . . . 129
A.1.2 Changed Specification Items in R26-11 . . . 130
A.1.3 Deleted Specification Items in R26-11 . . . 132
A.1.4 Added Constraints in R26-11 . . . 133
A.1.5 Changed Constraints in R26-11 . . . 133
A.1.6 Deleted Constraints in R26-11 . . . 133
A.2 Traceable item history of this document according to AUTOSAR Release R25-11 . . . 133
A.2.1 Added Specification Items in R25-11 . . . 133
A.2.2 Changed Specification Items in R25-11 . . . 134
A.2.3 Deleted Specification Items in R25-11 . . . 137
A.2.4 Added Constraints in R25-11 . . . 138
A.2.5 Changed Constraints in R25-11 . . . 138

A.2.6 Deleted Constraints in R25-11 138

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**).

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_FO_TR_Glossary [1].

Abbreviation / Acronym:	Description:
Csm	Crypto Service Manager
Dcm	Diagnostic Communication Manager
Dem event	Diagnostic Event Manager event
ECUC	ECU configuration
IDS	Intrusion Detection System
IdsM	Intrusion Detection System Manager
IdsR	Intrusion Detection System Reporter
IF	Interface
NvM	Non-volatile memory
PduR	PDU Router
QSEv	Qualified Security Event
SecXT	Security Extract
Sem	Security Event Memory
SEv	On-board Security Event
SOC	Security Operation Center
StbM	Synchronized Time-Base Manager

Table 2.1: Acronyms and Abbreviation

Abbreviation / Acronym:	Description:
API	see [1] AUTOSAR_FO_TR_Glossary
BSW	see [1] AUTOSAR_FO_TR_Glossary
BswM	see [1] AUTOSAR_FO_TR_Glossary
CDD	see [1] AUTOSAR_FO_TR_Glossary
Classic Platform	see [1] AUTOSAR_FO_TR_Glossary
Dem	see [1] AUTOSAR_FO_TR_Glossary
DET	see [1] AUTOSAR_FO_TR_Glossary
ECU	see [1] AUTOSAR_FO_TR_Glossary
ID	see [1] AUTOSAR_FO_TR_Glossary
MCU	see [1] AUTOSAR_FO_TR_Glossary
NVRAM	see [1] AUTOSAR_FO_TR_Glossary
OEM	see [1] AUTOSAR_FO_TR_Glossary
PDU	see [1] AUTOSAR_FO_TR_Glossary
RTE	see [1] AUTOSAR_FO_TR_Glossary
SW-C	see [1] AUTOSAR_FO_TR_Glossary
TP	see [1] AUTOSAR_FO_TR_Glossary

Table 2.2: Reference to Acronyms and Abbreviation

Terms:	Description:
Authenticator/ Authentification/ Authenticity	Relevant information to a QSEv. It is optional data and is the result of a cryptographic primitive provided by the IdsM, using the Csm in conjunction with the crypto stack to provide integrity and authenticity of the QSEvs. Examples would be digital signatures or message authentication codes (MAC).
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.
CsmJobPrimitive	A primitive is an instance of a configured cryptographic algorithm realized in the Csm (Crypto module)
CsmJob	A Job is a configured Job in the Crypto module. Among others, it refers to a key, a cryptographic primitive and a reference channel.
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.
IdsM block state	State reported by the BswM via IdsM_BswM_StateChanged. The states are used to suspend the collection of security events.
IDS Message	Message which is send by the IdsM with the IDS protocol.
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.
IdsM Qualified Event Buffers	Buffer to store the qualified security Events.
IdsM Number Of Qualified Event Buffers	Number of qualified event buffers for qualified security events.
NULL Pointer	Pointer to invalid memory address or object.
Qualified Security Event (QSEv)	Events that have passed their corresponding filter chain and are sent to the configured sink.
Reporting Mode	The reporting mode defines the detail of information which are forwarded with the qualified Security Event.
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.
Smart Sensor	Reporting identity that informs the IdsM module about SEvs. It is capable to deliver additional data to the QSEv, e.g. timestamp, counter value.

Terms:	Description:
Sink	Destination of a QSEv. Depending on the configuration the QSEv can be persisted, propagated or both.
Timestamp	Relevant information to a SEv. It is optional data that is provided by a sensor or the IdsM to be able to bring security events into the order of occurence.
Timestamp Provider	Service or SW-Component which provides a TimeStamp. e.g. in CP Stbm.

Table 2.3: Technical Terms

3 Related documentation

3.1 Related Specification

AUTOSAR provides a General Specification on Basic Software modules [2], 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_PRS_IntrusionDetectionSystem [3]:** Specifies the communication protocol for the transmission of security events.
- **AUTOSAR_MOD_GeneralDefinitions [4]:** Standardized Security Events reported by AUTOSAR BSW
- **AUTOSAR_TPS_SecurityExtractTemplate [5]:** Specifies the Security Extract.

3.2 Input Documents & Related Standards and Norms

[1] Glossary
AUTOSAR_FO_TR_Glossary

[2] General Specification of Basic Software Modules
AUTOSAR_CP_SWS_BSWGeneral

[3] Specification of Intrusion Detection System Protocol
AUTOSAR_FO_PRS_IntrusionDetectionSystem

[4] Standardized M1 Models used for the Definition of AUTOSAR
AUTOSAR_FO_MOD_GeneralDefinitions

[5] Security Extract Template
AUTOSAR_FO_TPS_SecurityExtractTemplate

[6] Technical Report on Security Events Specification
AUTOSAR_FO_TR_SecurityEventsSpecification

[7] System Template
AUTOSAR_CP_TPS_SystemTemplate

[8] Diagnostic Extract Template
AUTOSAR_CP_TPS_DiagnosticExtractTemplate

4 Constraints and assumptions

[SWS_IdsM_CONSTR_00001] ⌈The Intrusion Detection System 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.

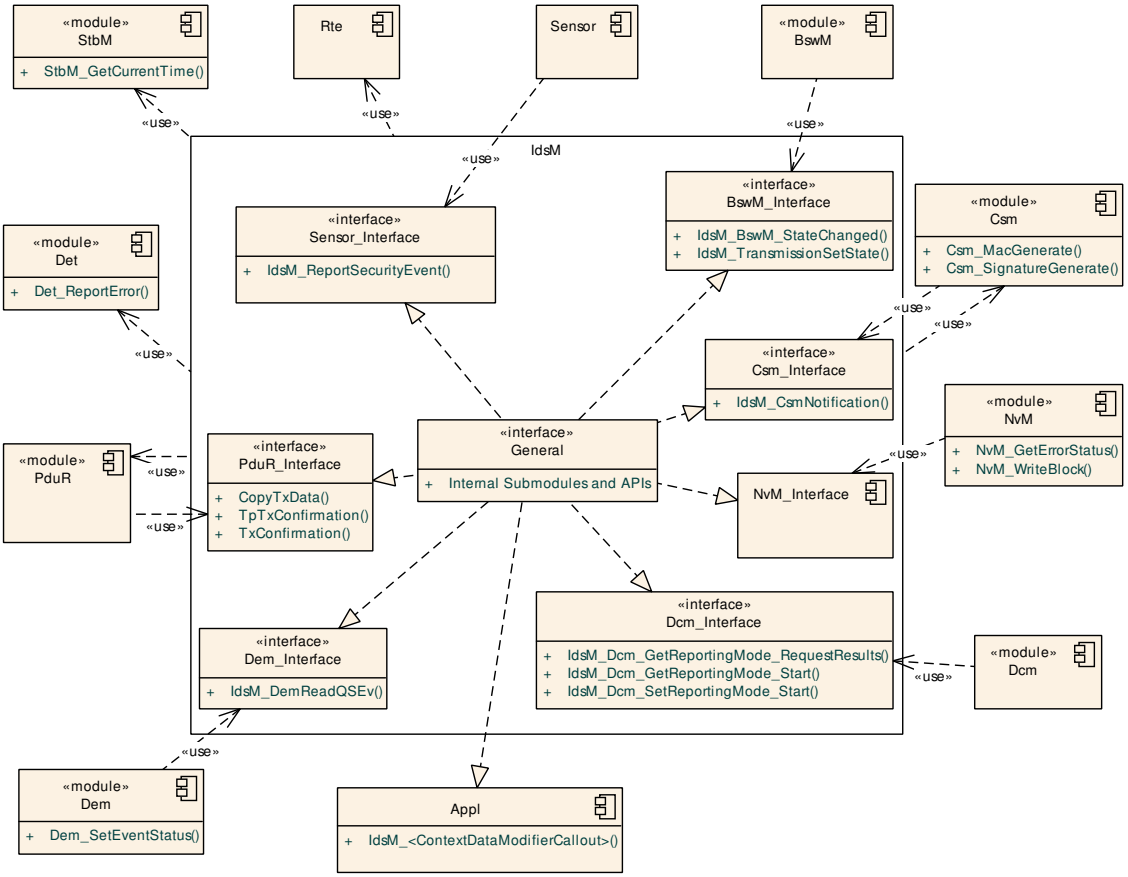


Figure 5.1: IdsM's interfaces to other modules

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 [2, SWS BSW General].

5.2.2 Header File Structure

Besides the files defined in section 5.1.7 “Header file structure” in [2, 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

This chapter is empty.

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:

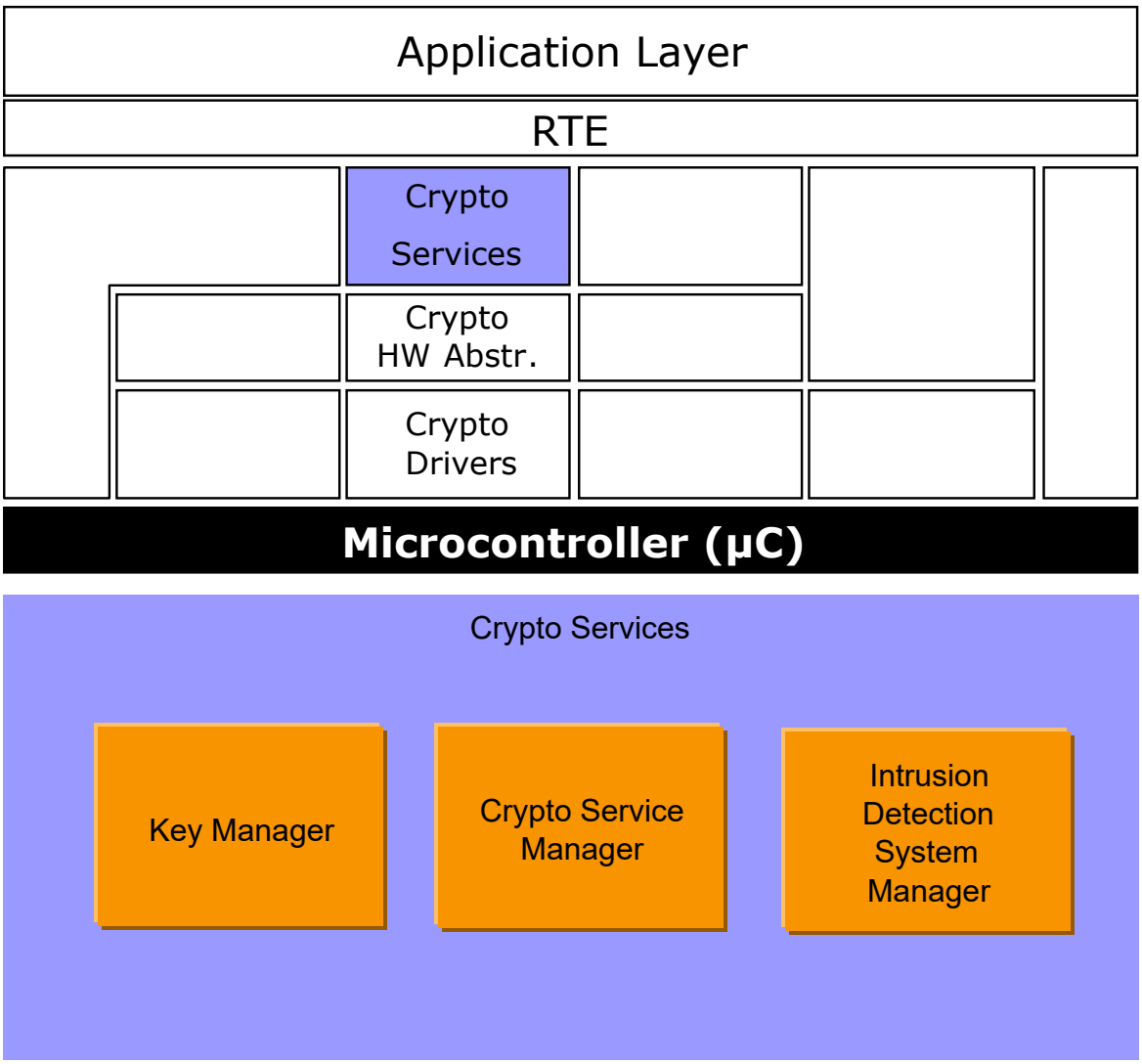


Figure 7.1: AUTOSAR BSW architecture showing the IdsM module

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

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, EthIf, KeyM and SecOC are illustrated as BSW sensor examples.

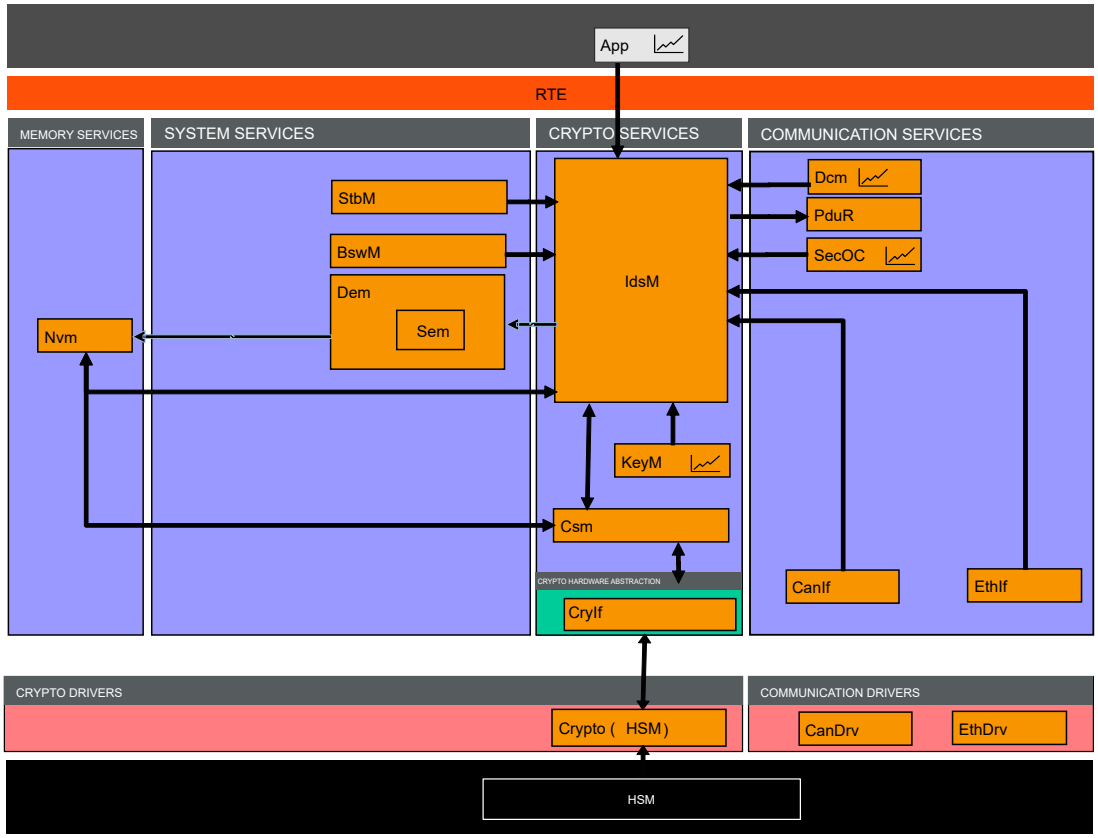


Figure 7.2: Interaction of the IdsM with other stack modules

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.

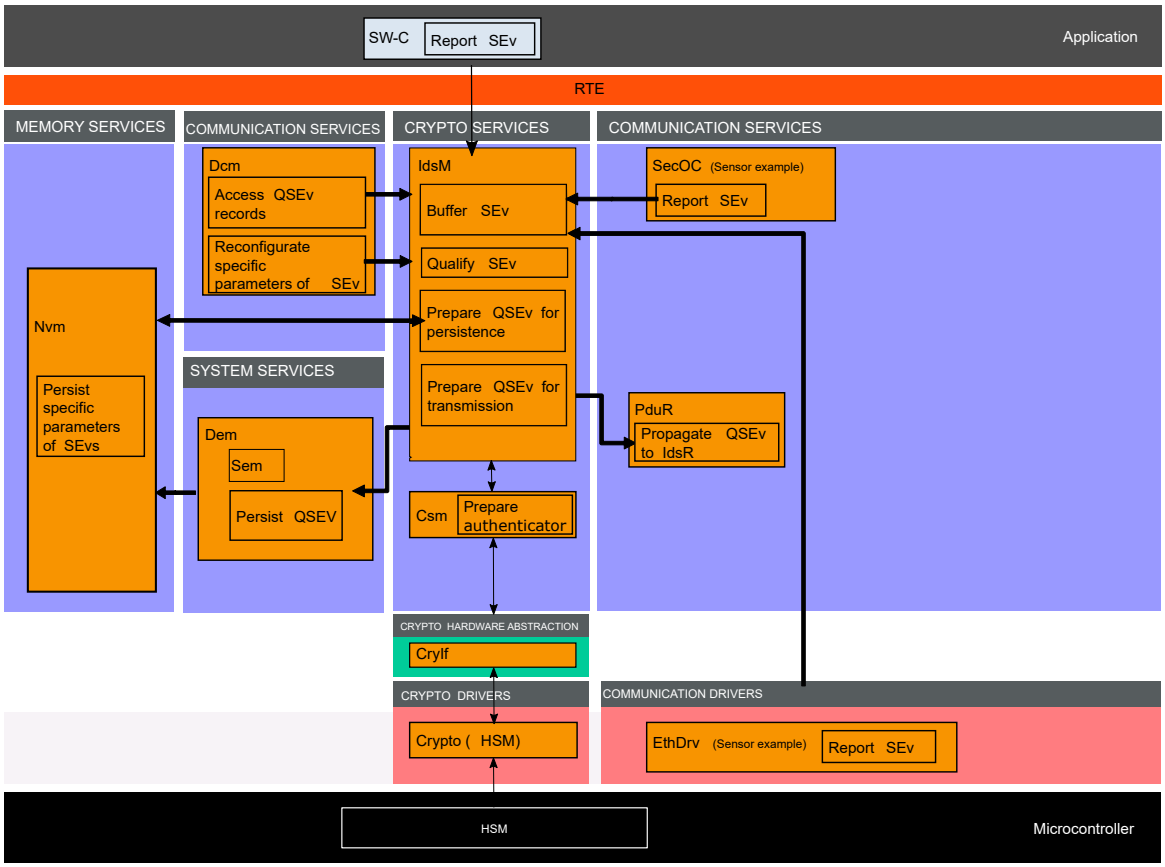


Figure 7.3: Functional modules of the IdsM.

7.2.1 Initialization

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

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

[SWS_IdsM_00203] Error UnInit ⌈If development error reporting is enabled via IdsMDevErrorDetect, the IdsM module shall call Det_ReportError with the error code IDSM_E_UNINIT when any API other than IdsM_Init is called in uninitialized state.⌋

[SWS_IdsM_00204] Error ReInit ⌈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_ReportSecurityEvent, 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_ReportSecurityEvent can be used by any sensor, independently of its source.

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.

[SWS_IdsM_00406] Reception of SEvs with optional parameters ⌈The IdsM shall be able to receive SEvs with additional optional parameters such as timestamp, counter calculated from a smart sensor and the versioned SEv context data, with the service IdsM_ReportSecurityEvent.

Note: For reporting SEvs with context data, the security sensor provides a context data pointer to a byte array where the context data is available, the corresponding size and version. The IdsM shall not rely on knowledge of the internal structure of the context data. For reporting a SEv with a timestamp but with no previously calculated counter, the security sensor provides the count value as 1.⌋

[SWS_IdsM_00503] Max Length Context Data ⌈IdsM 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 information, in that case it shall be evaluated that there are enough resources to avoid flooding of the communication channels.

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] Event Buffers configurable ⌈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] Event Buffer storage ⌈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] Availability of Event Buffer ⌈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 SEV_IDSM_NO_EVENT_BUFFER_AVAILABLE in [SWS_IdsM_91015].

[SWS_IdsM_00704] Context Data Buffers configurable ⌈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] Suitable Buffer Size ⌈Upon reception of a SEv with context data, the IdsM shall store the context data in a 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] Link Event to Context Data Buffer ⌈Once an appropriate Context Data Buffer has been found, it shall be linked to the corresponding Event Buffer for further processing.⌋

[SWS_IdsM_00707] No Suitable Buffer Size ⌈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] Availability of Context Data Buffer ⌈In case no appropriate Context Data Buffer is found. The IdsM internal SEV_IDSM_NO_EVENT_BUFFER_AVAILABLE shall be triggered, in case it has been configured.⌋

See SEV_IDSM_NO_CONTEXT_DATA_BUFFER_AVAILABLE in [SWS_IdsM_91015].

[SWS_IdsM_00709] SEv without Context Data ⌈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.⌋

[SWS_IdsM_00710] Reception SEv with higher priority ⌈Where IdsMEventDisplacementStrategy is IDSM_BUFFER_DISPLACEMENT_SEVERITY_BASED, when a new SEv is received and no suitable IdsMEventBuffers is found and the severity of the new SEv is higher than the severity of at least one SEv already in the buffer, IdsM shall replace the least severe SEv already in the buffer with the new SEv.

Note: In case where multiple events in the IdsMEventBuffers have the same least severity, it is left to the implementer which SEv to discard.

Example: Where the IdsMEventBuffers is 5 and all the buffers are full, when a new security event is reported, IdsM shall drop the least severe event amongst all the 6 events.

Rationale: Dropping a security event is inevitable when all the IdsMEventBuffers are full. In this case, dropping the least severe event minimizes the values of the security information lost.⌋

[SWS_IdsM_00711] Reception SEv with lower priority ⌈Where IdsMEventDisplacementStrategy is IDSM_BUFFER_DISPLACEMENT_SEVERITY_BASED, when a new SEv is received and no suitable IdsMEventBuffers is found and the severity of the new SEv is lower than or equal to the severity value of the least severe SEvs already in the buffer, IdsM shall drop the newly received SEv.⌋

[SWS_IdsM_00712] Configuration Drop Latest ⌈Where IdsMEventDisplacementStrategy is IDSM_BUFFER_DISPLACEMENT_DROP_LATEST, when a new SEv is received and no suitable IdsMEventBuffers is found, IdsM shall drop the newly received SEv.⌋

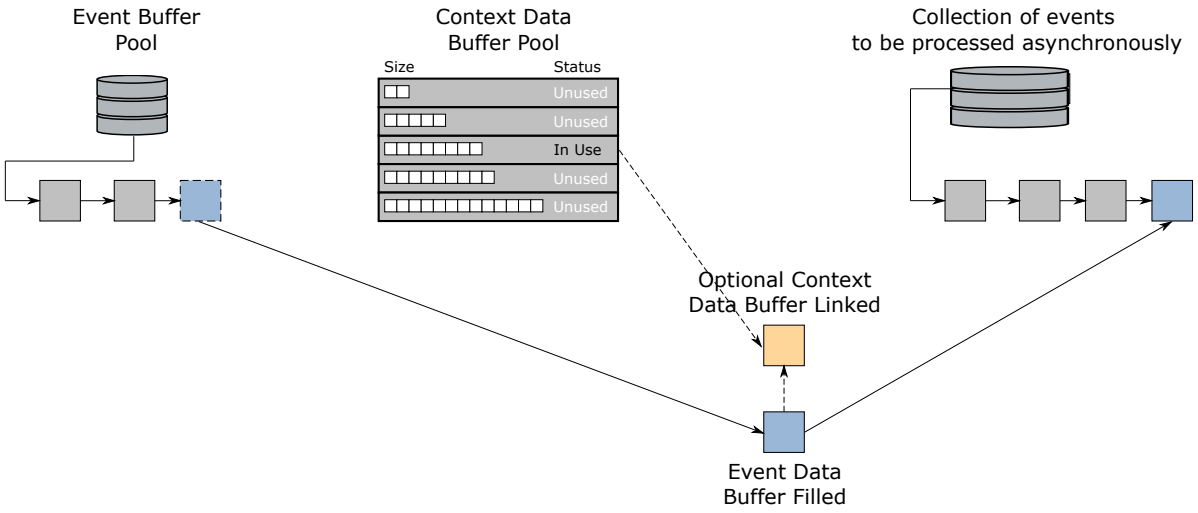


Figure 7.4: Use of Event Buffers and Context Data Buffers

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.

7.3.4 Context Data Modifier

An application shall be able to change the content of the originally reported context data of incoming SEvs.

If context data is to be modified, a combination of a IdsMContextDataModifierCallout and a IdsMContextDataModifierOptions can be configured.

The IdsMContextDataModifierCallout function is used to modify the context data and the contextDataSize is used to find a context data buffer of the adequate size.

[SWS_IdsM_00750] ⌈If the Context Data Modifier is configured, the IdsM shall call ContextDataModifierCallout directly after the corresponding SEv is reported, as long as the reporting mode is DETAILED or DETAILED_BYPASSING_FILTERS.

⌋

[SWS_IdsM_00751] ⌈If IdsM calls ContextDataModifierCallout, then it shall use the context data provided by the out parameter modifiedContextData as the SEv's context data.⌋

The IdsM can configure two kinds of context data modifiers:

- A specific modifier, IdsMContextDataModifierCallout, which affects only the SEvs linked to it.
- A global modifier, IdsMGlobalContextDataModifierCalloutFct, which affects all configured SEvs.

[SWS_IdsM_00752] ⌈If IdsMContextDataModifierCalloutFct is configured, SEvs shall call the configured callout function for the referenced SEvs. Otherwise, if IdsMGlobalContextDataModifierCalloutFct is configured, SEvs shall call the configured callout for all SEvs, except those with a specific modifier callout.⌋

[SWS_IdsM_00753] Append or trimming of Context Data ⌈For both global and the event specific Context Data Modifiers, IdsM shall support appending and trimming of context data. IdsM shall use the sign of the configured ContextDataSizeModifier to determine if the context data is to be appended to or trimmed.⌋

Below are some example scenarios to clarify the appending and trimming of context data.

Scenario 1: The security event ID = X is reported by a security sensor with N bytes of context data. The event has a modifiedContextDataconfigured with ContextDataSizeModifier as M (positive value).

	Reported Context Data Size	Configured ContextDataSizeModifier
SEv ID = X	N	M

Upon reception of this event, IdsM calls the configured IdsMContextData-ModifierCalloutFct with the following arguments:

uint16 securityEventId (in)	const uint8* contextData (in)	uint16 contextDataSize (in)	uint8* const modifiedContextDat (out)	uint16* const modifiedContext-DataSize (inout)
X	N bytes of originally reported context data	N	Buffer of size at least N+M	in: N+M out: P where P <= (N+M)

Where $(N+M) \in [0, 1500]$. When the function returns, IdsM is able to read modifiedContextDataSize bytes of data from modifiedContextData.

Scenario 2: The security event ID = Y is reported by a security sensor with N bytes of context data. The event has a Context Data modifier configured with modifiedContextData as -M (negative value).

	Reported Context Data Size	Configured ContextDataSizeModifier
SEv ID = Y	N	-M

Upon reception of this event, IdsM calls the configured IdsMContextDataModifierCalloutFct with the following arguments:

uint16 securityEvent-tId (in)	const uint8* contextData (in)	uint16 contextData-Size (in)	uint8* const modifiedContext-Data (out)	uint16* const modified-Context-DataSize (inout)
X	N bytes of originally reported context data	N	Buffer of size at least N-M	in: N-M out: P where P <= (N-M)

Where $(N + M) \in [0, 1500]$. When the function returns. IdsM is able to read modifiedContextDataSize bytes of data from modifiedContextData.

7.3.5 Context Data Endianess

SEv context data is generally structured (see e.g. [6]) and can contain multiple elements. On the IdsM side, the internal structure of the context data is not visible, as the context data is provided by the IDS sensor as a byte array.

In order to to fulfill [PRS_Ids_00004], all context data elements need to be provided in big endian byte order. This needs to be considered when constructing the SEv context data, i.e., the IDS sensor needs to take care of the endianess before submitting the SEv to the IdsM.

The API descriptions in Chapter 8 have been adapted accordingly to ensure that IDS sensors provide context data with correct byte order.

7.4 IdsM Internal SEvs

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

[SWS_IdsM_00801] Internal Security Events ⌈The security events reported by IdsM module are listed in [SWS_IdsM_91015].⌋

[SWS_IdsM_91015] Security events for IdsM ⌈

Name	*Description*	*ID*
SEV_IDSM_NO_EVENT_BUFFER_AVAILABLE	A SEv cannot be handled because there are no more event buffers available to process the event.	46
SEV_IDSM_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
SEV_IDSM_COMMUNICATION_ERROR	An error occurred when sending a QSEv via PDU.	49
SEV_IDSM_NO_QUALIFIED_EVENT_BUFFER_AVAILABLE	A security event raised when a QSEv has to be dropped due to insufficient QSEv buffers available.	87
SEV_IDSM_STATISTICS	A SEV reported by application to snapshot the IdsM statistics data.	126

⌋

[SWS_IdsM_00802] Independent Buffers ⌈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. Each of these buffers should store a single SEv.⌋

Note: Having dedicated buffers storing a single SEv each allows the IdsM to inform the sink about malfunctioning even if the IdsM is overloaded.

[SWS_IdsM_00803] No Instance Filtering ⌈IdsM internal SEvs shall not be filtered by IdsM instance specific filters.⌋

See 7.5 for filter categories.

[SWS_IdsM_00804] Specific Filterimg ⌈IdsM internal SEvs can be filtered by IdsM SEvID specific filters.⌋

See 7.5 for filter categories.

[SWS_IdsM_00805] Security Event Reporting ⌈If security event reporting is enabled in IdsM IdsMEnableSecurityEventReporting equals TRUE. IdsM shall monitor and raise the internal Security Events defined in [SWS_IdsM_91015].⌋

[SWS_IdsM_00806] PduR Transmit failed ⌈The IdsM internal SEv, SEV_IDSM_COMMUNICATION_ERROR , shall be raised when transmission request over PduR_IdsMTransmit was not accepted, i.e, return value of PduR_IdsMTransmit was E_NOT_OK.⌋

An example for this use case: The underlying bus in which the transmission was requested is unavailable and therefore the request was rejected.

[SWS_IdsM_00807] Negative transmission ⌈The IdsM internal SEv, SEV_IDSM_COMMUNICATION_ERROR, shall be raised when a negative transmission confirmation is delivered IdsM over IdsM_TxConfirmation or IdsM_TpTxConfirmation.⌋

Note: Forwarding the SEV_IDSM_COMMUNICATION_ERROR to the SinkIdsR might lead to recursive communication errors being detected.

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] Security Events stored in Event Buffer ⌈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] Qualification by Filtering ⌈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] Sampling and Aggregation ⌈A filter shall be able to modify the SEvs counter according to their algorithm, if they are of type sampling or aggregation.⌋

[SWS_IdsM_00907] ⌈If no filter in the configured filter chain dropped the SEv, it is considered a qualified security event (QSEv) and shall be stored in the Qualified Event Buffer (IdsMQualifiedEventBuffers).⌋

Note: The form in which the QSEv is buffered in IdsMQualifiedEventBuffers is up to the implementation, e.g. the QSEv could be packed according to [SWS_IdsM_01200] and buffered in IdsMQualifiedEventBuffers.

[SWS_IdsM_00908] ⌈IdsM shall have a configurable number of qualified event buffers (IdsMNumberOfQualifiedEventBuffers).⌋

[SWS_IdsM_00909] Internal Security Event No Qualified Event Buffer Available ⌈In case no appropriate qualified event buffers are found, the IdsM internal SEv SEV_IDSM_NO_QUALIFIED_EVENT_BUFFER_AVAILABLE shall be triggered in case it has been configured.⌋

[SWS_IdsM_00910] ⌈The QSEv in the qualified event buffer shall be forwarded to the corresponding sink depending on the IdsMEvent configuration.⌋

Note: Requirements in chapter 7.8 of SEvs shall be respected while forwarding to the sinks.

[SWS_IdsM_00911] Replace Last Severe QSEv ⌈Where IdsMEventDisplacementStrategy is IDSM_BUFFER_DISPLACEMENT_SEVERITY_BASED, when a new QSEv is generated and no suitable IdsMQualifiedEventBuffers is found and the severity of the new QSEv is higher than the severity of at least one QSEv already in the buffer, IdsM shall replace the least severe QSEv already in the buffer with the new QSEv.

Note: In case where multiple events in the IdsMQualifiedEventBuffers have the same least severity, it is left to the implementer which QSEv to discard.

Example: Where the IdsMQualifiedEventBuffers is 5 and all the buffers are full, when a new qualified security event is generated, IdsM shall drop the least severe event amongst all the 6 events.

Rationale: Dropping a qualified security event is inevitable when all the IdsM QualifiedEventBuffers are full. In this case, dropping the least severe event minimizes the values of the security information lost.

Note: IdsM internal SEv SEV_IDSM_NO_QUALIFIED_EVENT_BUFFER_AVAILABLE could be set to a higher severity to prevent a snowball effect.⌋

[SWS_IdsM_00912] Drop lower sever QSEv ⌈Where IdsMEventDisplacementStrategy is IDSM_BUFFER_DISPLACEMENT_SEVERITY_BASED, when a new QSEv is generated and no suitable IdsMQualifiedEventBuffers is found and the severity of the new QSEv is lower than or equal to the severity value of the least severe QSEv already in the buffer, IdsM shall drop the newly generated QSEv.⌋

[SWS_IdsM_00913] Drop latest received SEv ⌈Where IdsMEventDisplacementStrategy is IDSM_BUFFER_DISPLACEMENT_DROP_LATEST, when a new QSEv is generated and no suitable IdsMQualifiedEventBuffers is found, IdsM shall drop the newly generated QSEv.⌋

[SWS_IdsM_00914] SEv count overflow ⌈Filter algorithms shall not cause the SEv count to overflow. If the maximum value is reached, the value shall not be incremented any further.⌋

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.⌋

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

[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

⌋

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.⌋

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

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*

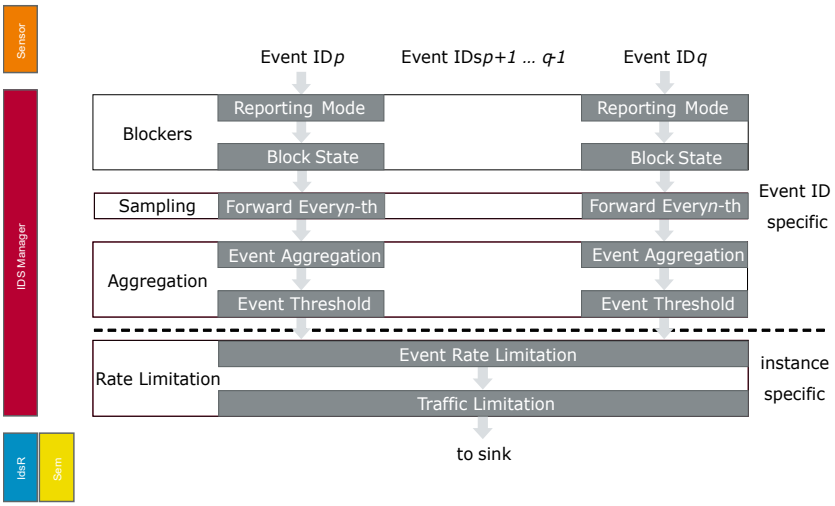


Figure 7.5: Filter categories and processing order

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] Mandatory for each Security Event ⌈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] Link to Security Event ⌈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.⌋

[SWS_IdsM_01013] Reporting Mode Options ⌈

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.

IdsM shall handle a reported SEv depending on its reporting mode according to this table

⌋

Table in [SWS_IdsM_01013] 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 [3]. The reporting mode is independent of the *Configuration Features*: **Timestamp and Authenticator.**

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 informed, 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.⌋

[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.⌋

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.⌋

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.

[SWS_IdsM_01034] ⌈If the IdsM receives a SEv with a count greater than 1 via the functions IdsM_ReportSecurityEvent, the aggregated count value of the filter may be greater than the configured threshold n. In this case, the SEv shall be forwarded containing the aggregated count value that may exceed the configured threshold n.⌋

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.⌋

[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 IdsMEvent-AggregationTimeInterval, 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.⌋

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.⌋

[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.⌋

[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.⌋

[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.⌋

[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.⌋

[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).⌋

[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:

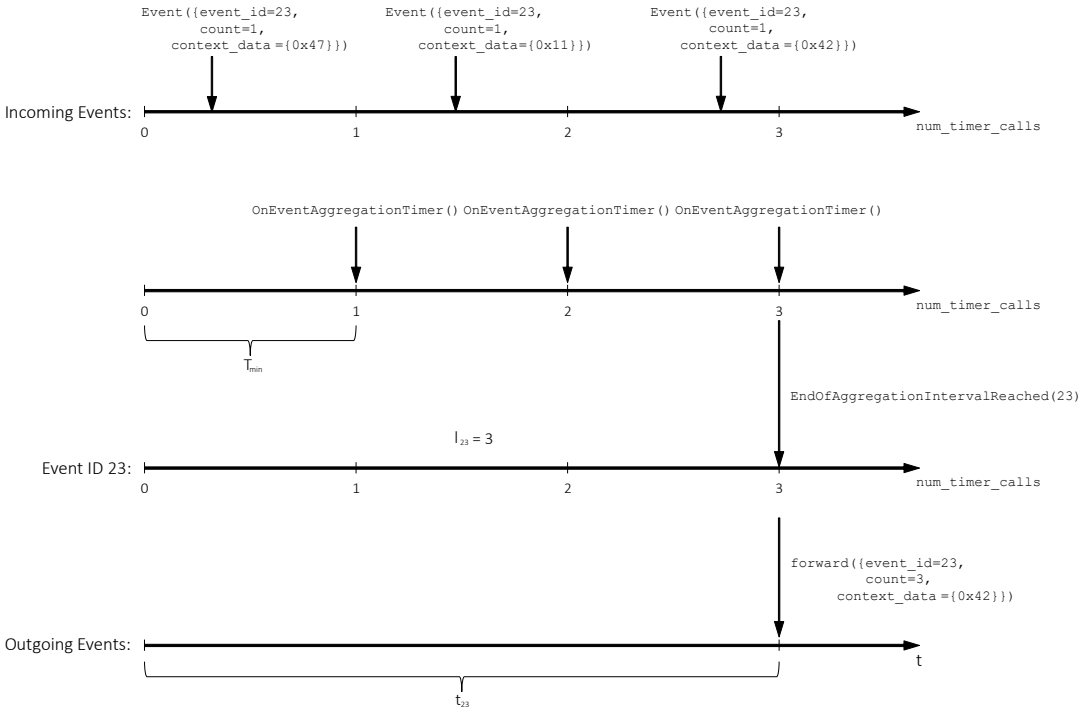


Figure 7.6: Example of aggregation filter

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 SEvs of given type that were processed by the event threshold filter in the current threshold interval is smaller than the configured parameter IdsMEventThresholdNumber.⌋

[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 SEvs 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.⌋

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:

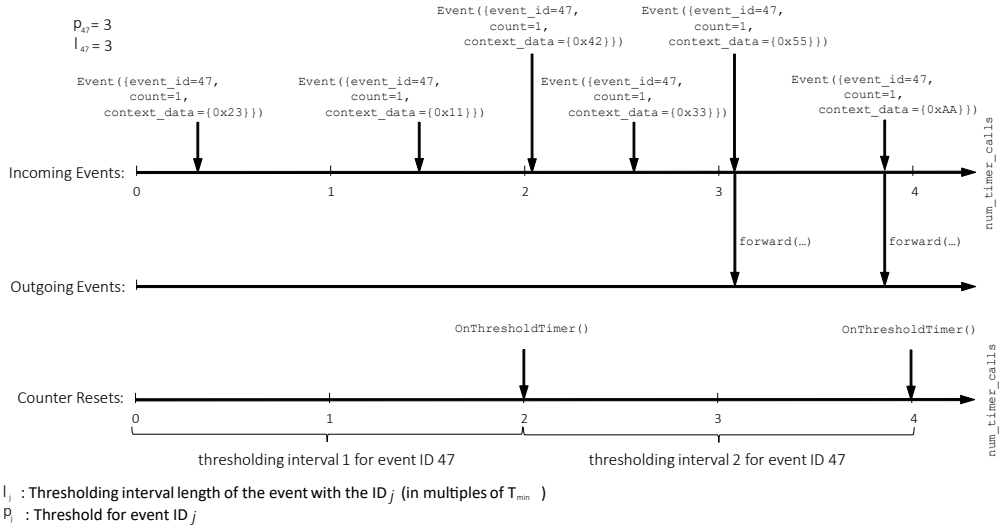


Figure 7.7: Example of threshold filter

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`.⌋

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 a 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.⌋

[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 a 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.⌋

[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.⌋

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 [3].

[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_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 [3].

[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_RequestCustomTimestamp 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 IdsMTimestampSupport equals FALSE, the timestamp feature is disabled and IdsM shall not add a timestamp to a QSEv. The IdsM shall ignore timestamps provided via the timestamp parameter of the event reporting interface.⌋

[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.⌋

[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_ReportSecurityEvent, then IdsM shall use this provided timestamp parameter for transmission or storage of the QSEv.⌋

[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, then IdsM shall set the option bit for the timestamp in protocol header to 0 before the transmission or storage of the QSEv.⌋

The below table Table 7.1 summarizes the presence and the source of the timestamp in the IDS message.

Timestamp reported	**IdsMTimestamp**	**IdsMTimestamp-Support**	Timestamp included in the IDS message	Requirement trace
0 - Timestamp not provided by sensor over SEv reporting API 1 - Timestamp provided over SEv reporting API	0 - The container IdsMTimestamp is not configured 1 - The container IdsMTimestamp is configured to use AR or custom timestamp source	0 - The parameter IdsMTimestampSupport is set to FALSE 1 - The parameter IdsMTimestampSupport is set to TRUE		
0	0	0	Not included, protocol header bit reset	[SWS_IdsM_01106]
0	0	1 (Default)	Not included, protocol header bit reset	[SWS_IdsM_01112]
0	1	0	Not included, protocol header bit reset	[SWS_IdsM_01106]
0	1	1	Yes. IDS message contains the timestamp fetched from the configured IdsMTimestampOption. The timestamp source coded as "Custom" or "AUTOSAR" depending on IdsMTimestampOption.	[SWS_IdsM_01111]
1	0	0	Not included, protocol header bit reset	[SWS_IdsM_01106]
1	0	1	Yes. IDS message contains the timestamp received over the SEv reporting interface. The timestamp source is coded as "Custom".	[SWS_IdsM_01108]
1	1	0	Not included, protocol header bit reset	[SWS_IdsM_01106]
1	1	1	Yes.	

Table 7.1: Timestamp Support

IDS message contains the timestamp received over the SEv reporting interface. The timestamp source is coded as "Custom" [SWS_IdsM_01108].

[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.⌋

[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.⌋

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

It is possible that the report event function is called in an order that does not match with the timestamp provided, i.e., the later call contains an older timestamp. This means that the persisted and transmitted events may contain timestamps that are not necessarily ordered.

[SWS_IdsM_01113] Timestamp Support ⌈If the parameter IdsMTimestampSupport equals TRUE, the timestamp feature is enabled. Note: It is not mandatory to configure IdsMTimestampOption when IdsMTimestampSupport is TRUE.⌋

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] ⌈

QSEv component	*Description*
Protocol Version and Header.	IdsM Protocol specific fields described in the [3, 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.
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.
Authenticator	Optional. Contains the authenticator of the QSEv, calculated over the complete IdsM message.

The QSEvs shall have a defined structure independent of the sink it is being sent to. The components of a QSEv are listed in this table

⌋

For further details of the IdsM Message structure refer to the [3, Specification of Intrusion Detection System Protocol].

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

[SWS_IdsM_01213] New Reporting API ⌈IdsM shall use the newest IDS protocol version to generate the QSEv for security events that were reported with the service IdsM_ReportSecurityEvent.⌋

7.8.2 Propagation of QSEvs: IdsR Sink

[SWS_IdsM_01202] Forwarding Security Events ⌈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 [3, 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.⌋

[SWS_IdsM_01209] Enable/Disable QSEv Transmission ⌈IdsM shall support enabling and disabling the forwarding of qualified security events to the sink IdsR during run-time over the API IdsM_TransmissionSetState .⌋

[SWS_IdsM_01210] Transmission State ⌈The IdsM shall be informed by the BswM module about the transmission state with the callback service IdsM_TransmissionSetState. This information shall be used when a QSEv is generated for transmission.⌋

[SWS_IdsM_01211] Transmission ON ⌈IdsM shall forward QSEvs to the sink IdsR only when the transmission is enabled IDSM_TRANSMISSION_STATE_ON.⌋

7.8.2.1 Authenticity of QSEvs

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

[SWS_IdsM_01204] Authenticating QSEv ⌈The IdsM shall be able to attach a cryptographic authenticator, with the same data format, to each QSEv. The authentication feature is optional and shall be activated or deactivated globally for all QSEvs by setting the parameter IdsMAuthenticatorSupport to TRUE or FALSE.⌋

[SWS_IdsM_01205] AuthenticatorLength ⌈The IdsM's parameter IdsMAuthenticatorLength shall define the length in bytes of the authenticator calculated by the crypto services. It shall be configured when the authentication feature is activated.⌋

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

[SWS_IdsM_01206] GenerateAuthenticator ⌈In order to generate an authenticator by a CsmJob IdsMCsmJobReference, the authentication generation shall be triggered by calling the Csm function Csm_SignatureGenerate or Csm_MacGenerate depending on the CsmJobPrimitive referenced by the CsmJob.⌋

[SWS_IdsM_01207] SynchronousAuthenticatorGeneration ⌈If the authenticator is generated by a synchronous Csm job IdsMCsmJobReference, when the function Csm_SignatureGenerate or Csm_MacGenerate returns, the authenticator shall be immediately available.⌋

[SWS_IdsM_01208] AsynchronousAuthenticatorGeneration ⌈If the authenticator is generated by an asynchronous CsmJob IdsMCsmJobReference, the IdsM shall be informed about the generation of the authenticator 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 CsmJobPrimitive callback for the CsmJob configured for the IdsM.

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

Over which data the authenticator shall be computed and how the authenticator shall be included in the IDS Message Structure, is specified in [3, 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 report additional information to the IdsM. The necessary configurations can be made per SEv via the parameters in container IdsMServiceInterfaceOptions.

[SWS_IdsM_01300] Context Data Size Limit ⌈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_01304] NULL-Pointer valid for parameter for ContextData-Pointer ⌈For an IdsMExternalEventId, when IdsMEventMaxContextDataSize is zero and the IdsMExternalEventId is in range 0x8000 - 0xFFFE, IdsM shall offer a service interface for reporting the security event with a possibility to provide NULL_PTR to the contextData argument as specified in [SWS_IdsM_91044] IdsMService_{EventName}⌋

[SWS_IdsM_01305] NULL-Pointer valid for parameter for Timestamp-Pointer ⌈For an IdsMExternalEventId, when IdsMEventEnableTimestampReporting is False and the IdsMExternalEventId is in range 0x8000 - 0xFFFE, IdsM shall offer a service interface for reporting the security event with a possibility to provide NULL_PTR to the timestamp argument as specified in [SWS_IdsM_91044] IdsMService_{EventName}⌋

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 Authenticator (optional)

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

[SWS_IdsM_01401] Transport Protocol ⌈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] Pdu Transmission ⌈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] Consider Tx Confirmation before next Transmit ⌈IdsM shall not call PduR_IdsMTransmit again before IdsM_TpTxConfirmation or IdsM_TxConfirmation have been called.⌋

[SWS_IdsM_01500] Using Tx Confirmation ⌈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] Segmented Transmission ⌈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] Using Tx Confirmation for Segmented Transmission ⌈The IdsM shall receive the confirmation of the transmission of a segmented PDU by the PduR Module with the service IdsM_TpTxConfirmation.⌋

7.8.2.4 Transmission Retry Handling

[SWS_IdsM_01503] Retry Handling ⌈If the transmission request over PduR_IdsMTransmit is declined, i.e., the API does not return E_OK, IdsM shall wait a time period of IdsMTransmissionRetryDelay and attempt to transmit the same message again.⌋

Note: The retry mechanism applied for IF and TP Pdus.

7.8.3 Storage of Events: Dem / Sem Sink

In order to use Dem as a sink to IdsM, the user must provide sufficient configuration, namely, IdsMDemEventReference and IdsMSinkDem. The Dem event referenced over IdsMDemEventReference is configured to have a callback to IdsM_DemReadQSEv and sufficient freeze frame size to accept the QSEv as specified in the IDS message. Upon qualification of a security event, IdsM notifies a FAILED event to Dem by calling the Dem_SetEventStatus and Dem subsequently calls IdsM over IdsM_DemReadQSEv to fetch the IDS message. To conclude the storage of the event in Dem, IdsM calls Dem_SetEventStatus again and reports the same event as PASSED. The FAILED report to Dem, the callback to IdsM and the PASSED report to Dem indicate the storage of a singular qualified security event. At most one concurrent storage of qualified security event in Dem is supported.

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

[SWS_IdsM_01601] ⌈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.⌋

[SWS_IdsM_01603] Dem Event referenced by IdsM Event ⌈IdsMEvent shall have a mapping to DemEventParameter via configuration parameter IdsMDemEventReference. Every IdsMEvent that is intended to be sinked to the Dem / Sem shall have a valid IdsMDemEventReference not necessarily unique.⌋

[SWS_IdsM_01604] Dem API ⌈If IdsMSinkDem for an IdsMEvent is set to true, then IdsM shall report to Dem by calling API Dem_SetEventStatus with arguments EventId as the DemEventId of the referenced DemEventParameter over IdsMDemEventReference and EventStatus as DEM_EVENT_STATUS_FAILED.⌋

Note: IdsM shall use AR standardized symbolic name for the DemEventId.

[SWS_IdsM_01605] Dem Callback With Freeze Frame ⌈IdsM shall provide complete IDS Message with its mandatory and optional fields as freeze frame data of the reported Dem event via a callback function IdsM_DemReadQSEv.⌋

[SWS_IdsM_01606] API Call Sequence ⌈IdsM shall call Dem_SetEventStatus at most once before IdsM_DemReadQSEv has been called by Dem to fetch the freeze frame data. This is necessary for Dem to fetch the correct QSEv as the freeze frame record.⌋

Note: Since the storage of the qualified security events in Dem is asynchronous and can span over multiple main function cycles, the IdsMQualifiedEvent Buffers can be used to queue the subsequently qualified events.

[SWS_IdsM_01607] Finish Call Sequence ⌈Once Dem has fetched the freeze frame data, IdsM shall call Dem_SetEventStatus with EventStatus as DEM_EVENT_STATUS_PASSED.⌋

(Refer to: 9.1 Sequence Diagram IdsM DEM.)

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 IdsMReporting-ModeFilter has been successfully changed by the diagnostic services.⌋

The modification of a "Reporting Mode" value is described in 7.10.

[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 IdsMNvmBlock-Descriptor, 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 Reconfiguration of 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 via Diagnostics

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

[SWS_IdsM_01901] Modify Reporting Mode via Diagnostics ⌈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] Invalid Reporting Mode Parameter ⌈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] Invalid Reporting Mode Parameter ⌈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] Persist Failure ⌈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] Roll Back in case of Failure ⌈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.⌋

[SWS_IdsM_02000] Readable Reporting Mode ⌈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] Reading Reporting Mode ⌈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] Reporting Mode Result ⌈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.⌋

7.10.2 Reconfiguration via Reporting Mode API

To enable access to "Reporting Mode" from SWCs the Reporting Mode API is provided by IdsM.

[SWS_IdsM_02030] Reporting Mode ⌈The "Reporting Mode" IdsMReportingModeFilter of a SEv shall be modifiable during run-time via the Reporting Mode API.⌋

[SWS_IdsM_02031] Modify Reporting Mode ⌈The service IdsM_SetReportingMode shall enable the IdsM to modify the reporting mode IdsMReportingModeFilter of a specific SEv.⌋

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_02032] Reading Reporting Mode ⌈The service IdsM_GetReportingMode reads out the Reporting Mode of a specific SEv.⌋

[SWS_IdsM_02033] Invalid Parameters ⌈In case one of the input parameters of IdsM_SetReportingMode or IdsM_GetReportingMode are invalid, this IdsM Service shall return IDSM_E_PARAM_INVALID.⌋

[SWS_IdsM_02034] Busy Failure ⌈In case the IdsM_SetReportingMode is called while an operation is active, this IdsM Service function shall return IDSM_E_BUSY.⌋

[SWS_IdsM_02035] Internal Error ⌈In case the IdsM_SetReportingMode or IdsM_GetReportingMode is called and encounters an internal error, this IdsM Service function shall return E_NOT_OK.⌋

[SWS_IdsM_02036] Polling of Set Reporting Mode Operation ⌈The Service IdsM_GetStatusReportingMode provides the option to poll the status of a set reporting mode operation. For operation status refer to 8.2.6.⌋

7.11 Statistics

IdsM maintains statistics of received security events, qualified security events and the peak fill level of all of its buffers. An event is considered received when it is provided over one of the reporting APIs. A qualified event is described in [SWS_IdsM_00907]. The peak fill status is the highest number of buffers filled as observed by the periodic IdsM_MainFunction.

[SWS_IdsM_03004] Global Count Security Events ⌈IdsM shall maintain a global count of the number of security events received when IdsMInstanceStatistics is set to TRUE.⌋

[SWS_IdsM_03005] Global Count Qualified Security Events ⌈IdsM shall maintain a global count of the number of security events qualified when IdsMInstanceStatistics is set to TRUE.⌋

[SWS_IdsM_03006] Global Count Transmitted Security Events ⌈IdsM shall maintain a global count of the number of security events transmitted (Sink IdsR) successfully when IdsMInstanceStatistics is set to TRUE.⌋

[SWS_IdsM_03007] Global Count Failed Transmitted Security Events ⌈IdsM shall maintain a global count of the number of security events failed to be transmitted (Sink IdsR) when IdsMInstanceStatistics is set to TRUE.⌋

[SWS_IdsM_03008] Global Count Persisted Security Events ⌈IdsM shall maintain a global count of the number of security events persisted (Sink Sem) when IdsMInstanceStatistics is set to TRUE.⌋

[SWS_IdsM_03009] Event Count Received Security Events ⌈IdsM shall maintain a event count of the number of security events received when IdsMInstanceStatistics is set to TRUE.⌋

[SWS_IdsM_03010] Event Count Qualified Security Events ⌈IdsM shall maintain a event count of the number of security events qualified when IdsMInstanceStatistics is set to TRUE.⌋

[SWS_IdsM_03011] Peak Fill Status Event Buffer ⌈IdsM shall maintain a peak fill status of the IdsMEventBuffers when IdsMInstanceStatistics is set to TRUE.⌋

[SWS_IdsM_03012] Peak Fill Status Size Context Data Buffer ⌈IdsM shall maintain a peak fill status of the IdsMContextDataBuffer when IdsMInstanceStatistics is set to TRUE.⌋

[SWS_IdsM_03013] Peak Fill Status Size Qualified Event Buffer ⌈IdsM shall maintain a peak fill status of the IdsMQualifiedEventBuffers when IdsMInstanceStatistics is set to TRUE.⌋

[SWS_IdsM_03014] Persist Statistics ⌈If IdsMPersistStatistics is TRUE. IdsM shall persist the statistics in non-volatile memory. e.g. the statistics collected after the latest IdsM initialization are added to the statistics that was persisted previously. The storage of the statistics in the non-volatile memory shall be implementation specific.⌋

[SWS_IdsM_03015] Add Statistics to Context Data ⌈If SEV_IDSM_STATISTICS is configured, when the referenced security event is reported, IdsM shall add the collected statistics as a context data to the event as described in [SWS_IdsM_03016], [SWS_IdsM_03017], [SWS_IdsM_03018], [SWS_IdsM_03019].⌋

Note: This security event is a special case where IdsM adds context data to an externally reported security event.

7.11.1 Configuration IdsM Statistics

[ECUC_IdsM_00093] Definition of EcucParamConfContainerDef IdsMStatistics ⌈

Container Name	IdsMStatistics
Parent Container	IdsMConfiguration
Upstream mapping defined	–
Description	Configuration parameters for statistics.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
IdsMInstanceStatistics	1	[ECUC_IdsM_00094]
IdsMPersistStatistics	1	[ECUC_IdsM_00095]
SEV_IDSM_STATISTICS	0..1	[ECUC_IdsM_00097]

No Included Containers

⌋

[ECUC_IdsM_00094] Definition of EcucBooleanParamDef IdsMInstanceStatistics ⌈

Parameter Name	IdsMInstanceStatistics
Parent Container	IdsMStatistics
Upstream mapping defined	–
Description	A boolean configuration to enable and disable collecting instance statistics.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

⌋

[ECUC_IdsM_00095] Definition of EcucBooleanParamDef IdsMPersistStatistics ⌈

Parameter Name	IdsMPersistStatistics
Parent Container	IdsMStatistics
Upstream mapping defined	–
Description	A boolean configuration to enable and disable local persistence of statistics.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

⌋

[ECUC_IdsM_00097] Definition of EcucReferenceDef SEV_IDSM_STATISTICS ⌈

Parameter Name	SEV_IDSM_STATISTICS
Parent Container	IdsMStatistics
Upstream mapping defined	–
Description	This parameter is a reference to the IdsMEvent to which IdsM shall add the statistics in the context data.
Multiplicity	0..1
Type	Reference to IdsMEvent
Pre-Compile only	true

⌋

7.11.2 Security Event IdsM Statistics

The `security event` `SEV_IDSM_STATISTICS` is defined with the `context data`.

[SWS_IdsM_03016] Security event context data definition: SEV_IDSM_STATISTICS

Status: DRAFT

⌈

SEV Name	***SEV_IDSM_STATISTICS***	
ID	126	
Description	A SEV reported by application to snapshot the IdsM statistics data.	
Context Data Version	1	
Context Data	***Data Type***	***Allowed Values***
ControlField	uint8	The 8 Bit control field is used to describe the content of the context data in the following bytes.
InstanceStatisticsForEvents	uint8 [10]	(Optional) The instance statistics segment contains various types of counts for events in an IdsM instance, e.g., number of events received, transmitted and stored. This segment of the context data is optional and its availability is indicated by Bit0 of the ControlField.
InstanceStatisticsForEvent Buffers	uint8 [4]	(Optional) The instance statistics segment contains buffer fill level information that are collected for an IdsM instance, e.g., peak fill level of the event and qualified event buffers. This segment of the context data is optional and its availability is indicated by Bit0 of the ControlField.
InstanceStatisticsContext DataBuffer	uint8 [380]	(Optional) The instance statistics segment contains the peakfill level of all the unique context data buffers configured in IdsM. The field has a variable length because the IdsM provides the possibility to configure a wide-range of context data buffers of different sizes. The variable length in bytes is equal to 2 + (6 x Number of unique context data buffer sizes). This segment of the context data is optional and its availability is indicated by Bit0 of the ControlField. Note: Since IdsM supports maximum of 1500 bytes of context data (ECUC_IdsM_00047), the number unique context data buffers for which statistics collected is limited to 63. The maximum length for this field is 380 bytes (2 + (63 * 6) bytes).
	MAX-LENGTH	Truncation length can be defined on a project specific basis. AUTOSAR has defined a default value, as given in the InstanceStatisticsContextDataBuffer byte array definition above.
EventSpecificStatistics	uint8 [891]	(Optional) The event specific statistics segment contains individual counts of reception and qualifcation for select events. This segment of the context data is optional and has a variable length because IdsM provides the possibility to configure IdsMEventStatistics for every IdsMEvent. The variable length in bytes is equal to 2 + (7 x Number of IdsMEvent with IdsMEventStatistics set to TRUE). This segment of the context data is optional and its availability is indicated by Bit1 of the ControlField. Note: Since IdsM support maximum of 1500 bytes of context data (ECUC_IdsM_00047), the number of events for which statistics collected is limited to 127. The maximum length for this field is 891 (2 + (127 * 7) bytes).
	MAX-LENGTH	Truncation length can be defined on a project specific basis. AUTOSAR has defined a default value, as given in the EventSpecificStatistics byte array definition above.

⌋

[SWS_IdsM_03017] Statistics Control Field ⌈The ControlField shall contain the control bits as shown below⌋

Control Field							
Byte 0							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0-Statistics are not persisted 1-Statistics are persisted	reserved	reserved	reserved	reserved	reserved	0-EventSpecific Statistics not included 1-EventSpecific Statistics included	0-Both InstanceStatistics segments not included 1-Both InstanceStatistics segments included

[SWS_IdsM_03018] Instance Statistics Buffers ⌈The InstanceStatisticsEventBuffers and InstanceStatisticsContextDataBuffer shall be structured as shown below⌋

InstanceStatisticsEvents Size= 2B+2B+2B+2B+2B (optional)									
Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9
Number of Events received		Number of Events qualified		Number of Events transmitted		Number of Events failed to transmit		Number of Events persisted	
[SWS_IdsM_03004]		[SWS_IdsM_03005]		[SWS_IdsM_03006]		[SWS_IdsM_03007]		[SWS_IdsM_03008]	

InstanceStatisticsEventBuffers Size= 2B+2B (optional)			
Byte 0	Byte 1	Byte 2	Byte 3
Peak fill level of event buffers		Peak fill level of qualified event buffers	
[SWS_IdsM_03011]		[SWS_IdsM_03013]	

InstanceStatisticsContextDataBuffer Size= 2B+(6B*N Unique Context Data Buffer Sizes) (optional)														
Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10	Byte 11	Byte 12	Byte 13	6 bytes * (n-2) unique context data buffers
Number of unique context data buffers		#1 context data buffer size		#1 context data buffer count		#1 Peak fill level of context data buffer		#2 context data buffer size		#2 context data buffer count		#2 Peak fill level of context data buffer		
[SWS_IdsM_03012]														

[SWS_IdsM_03019] Event Specific Statistics ⌈The EventSpecificStatistics shall be structured as shown below⌋

EventSpecificStatistics Size= 2B+(7B*N Events) (optional)																
Byte 0	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**	**Byte 8**	**Byte 9**	**Byte 10**	**Byte 11**	**Byte 12**	**Byte 13**	**Byte 14**	**Byte 15**	**7 bytes * (n-2) events**
Number of unique event statistics in the segment		Sensor Instance ID	Event Definition ID		Number of events received		Number of events qualified		Sensor Instance ID	Event Definition ID		Number of events received		Number of events qualified		. . . .
[SWS_IdsM_03009]; [SWS_IdsM_03010]																

[SWS_IdsM_03020] Drop ⌈IdsM shall drop any context data reported by application over the reporting APIs for the IdsMEvent referenced by SEV_IDSM_STATISTICS⌋

7.12 Error Classification

7.12.1 Development Errors

[SWS_IdsM_02003] Definition of development errors in module IdsM ⌈

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

⌋

7.12.2 Runtime Errors

The IdsM module does not define runtime errors.

7.12.3 Production Errors

The IdsM module does not define production errors.

7.13 Error Detection and Notification

For details about error detection and notification of BSW modules refer to [2] Chapter 7.2 *“Error Handling”*.

7.13.1 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].

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

[SWS_IdsM_02110] Unknown Event ⌈With development error detection IdsMDevErrorDetect enabled, the IdsM module shall check the parameter securityEventId of the function IdsM_ReportSecurityEvent 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.c⌋

[SWS_IdsM_02111] Size Exceed ⌈With development error detection IdsMDevErrorDetect enabled, the IdsM module shall check the parameter contextDataSize of the function IdsM_ReportSecurityEvent, 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_02112] Wrong Count ⌈With development error detection IdsMDevErrorDetect enabled, the IdsM module shall check the count passed over the function IdsM_ReportSecurityEvent. The development error IDSM_E_PARAM_INVALID shall be reported when the passed count is equal to 0.⌋

[SWS_IdsM_02113] Wrong Context Data Version ⌈With development error detection IdsMDevErrorDetect enabled, the IdsM module shall check the context data version passed over the function IdsM_ReportSecurityEvent. The development error IDSM_E_PARAM_INVALID shall be reported when the passed version is equal to 0.⌋

8 API specification

8.1 Imported Types

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

[SWS_IdsM_91022] Definition of imported datatypes of module IdsM ⌈

Module	*Header File*	*Imported Type*
Comtype	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
Csm	Rte_Csm_Type.h	Crypto_OperationModeType
	Rte_Csm_Type.h	Crypto_ResultType
Dcm	Rte_Dcm_Type.h	Dcm_NegativeResponseCodeType
	Rte_Dcm_Type.h	Dcm_OpStatusType
NvM	Rte_NvM_Type.h	NvM_BlockIdType
	Rte_NvM_Type.h	NvM_RequestResultType
StbM	Rte_StbM_Type.h	StbM_SynchronizedTimeBaseType
	Rte_StbM_Type.h	StbM_TimeBaseStatusType
	Rte_StbM_Type.h	StbM_TimeStampType
	Rte_StbM_Type.h	StbM_TimeTupleType
	Rte_StbM_Type.h	StbM_UserDataType
	StbM.h	StbM_VirtualLocalTimeType
Std	Std_Types.h	Std_ReturnType

⌋

8.2 Type Definitions

8.2.1 IdsM_ConfigType

[SWS_IdsM_91012] Definition of datatype IdsM_ConfigType ⌈

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_SecurityEventIdType

[SWS_IdsM_91031] Definition of ImplementationDataType IdsM_SecurityEventIdType ⌈

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.h		

⌋

8.2.3 IdsM_SensorInstanceIdType

[SWS_IdsM_91049] Definition of ImplementationDataType IdsM_SensorInstanceIdType ⌈

Name	IdsM_SensorInstanceIdType		
Kind	Type		
Derived from	uint8		
Range	0..63	–	–
Description	Data type for IdsM reporting mode		
Variation	–		
Available via	Rte_IdsM_Type.h		

⌋

8.2.4 IdsM_Filters_BlockStateType

[SWS_IdsM_91017] Definition of datatype IdsM_Filters_BlockStateType ⌈

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.h		

⌋

8.2.5 IdsM_Filters_ReportingModeType

[SWS_IdsM_91013] Definition of ImplementationDataType IdsM_Filters_ReportingModeType ⌈

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		
Variation	–		
Available via	Rte_IdsM_Type.h		

⌋

8.2.6 IdsM_ReportingModeStatusType

[SWS_IdsM_91048] Definition of ImplementationDataType IdsM_ReportingModeStatusType ⌈

Name	IdsM_ReportingModeStatusType		
Kind	Type		
Derived from	uint8		
Range	IDSM_REPORTING_MODE_STATUS_NO_OPERATION	0x00	No operation in progress
	IDSM_REPORTING_MODE_STATUS__SUCCESSFUL	0x01	Operation successful
	IDSM_REPORTING_MODE_STATUS__PENDING	0x02	Operation still in progress
	IDSM_REPORTING_MODE_STATUS__FAILED	0x03	Operation failed
Description	Data type for IdsM reporting mode		
Variation	–		
Available via	Rte_IdsM_Type.h		

⌋

8.2.7 IdsM_ReturnType

[SWS_IdsM_91050] Definition of Std_ReturnType-extension for module IdsM ⌈

Range	IDSM_E_BUSY	0x03	The service request failed because the service is still busy
	IDSM_E_PARAM_INVALID	0x0A	The service request failed because of invalid input arguments
Description	–		
Available via	Rte_IdsM_Type.h		

⌋

8.2.8 IdsM_TransmissionStateType

[SWS_IdsM_91035] Definition of datatype IdsM_TransmissionStateType ⌈

Name	IdsM_TransmissionStateType		
Kind	Type		
Derived from	uint8		
Range	IDSM_TRANSMISSION_STATE_OFF	0x00	–
	IDSM_TRANSMISSION_STATE_ON	0x01	–
Description	Data type used for transmission state in the API IdsM_TransmissionSetState.		
Available via	IdsM.h		

⌋

8.3 Function Definitions

This chapter contains all APIs provided by the module.

8.3.1 IdsM_Init

[SWS_IdsM_91001] Definition of API function IdsM_Init ⌈

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_ReportSecurityEvent

[SWS_IdsM_91038] Definition of API function IdsM_ReportSecurityEvent ⌈

Service Name	IdsM_ReportSecurityEvent	

Syntax	`void IdsM_ReportSecurityEvent (`
	`  IdsM_SecurityEventIdType securityEventId,`
	`  const uint8* contextData,`
	`  uint16 contextDataSize,`
	`  uint16 contextDataVersion,`
	`  uint16 count,`
	`  const IdsM_TimestampDataType* timestamp`
	`)`	

Service ID [hex]	0x13	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	securityEventId	Security Event ID. Symolic name of the configured SEvs.
	contextData	Pointer to context data in big endian format. Use NULL_PTR if no context data is available.
	contextDataSize	Size of context data. If no context data is available, provide 0.
	contextDataVersion	Version of context data. If no context data is available, provide 1.
	count	Number of occurrences of the security events since last report. Permitted range [1, max(uint16)]. If sensor didn't maintain a special count, provide 1.
	timestamp	Pointer to timestamp used for time reference of the security event. Use NULL_PTR if no timestamp is available.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	This API is the interface to report security events to the IdsM.	
Available via	IdsM.h	

⌋

Please check chapter 7.3.5 for more information about context data endianess.

8.3.3 IdsM_GetReportingMode

[SWS_IdsM_91051] Definition of API function IdsM_GetReportingMode ⌈

Service Name	IdsM_GetReportingMode	
Syntax	`Std_ReturnType IdsM_GetReportingMode ( IdsM_SecurityEventIdType securityEventId, IdsM_SensorInstanceIdType sensorInstanceId, IdsM_Filters_ReportingModeType* reportingMode )`	
Service ID [hex]	0x14	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	securityEventId	External ID of the Security Event from whom the reporting mode shall be returned
	sensorInstanceId	ID of the sensor instance of the security event
	reportingMode	The reporting mode for the requested Security Event
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: Request successful `E_NOT_OK`: Request failed `IDSM_E_PARAM_INVALID`: Request failed since invalid input argument `IDSM_E_BUSY`: Request failed since service is busy
Description	This API is the application interface to get the current reporting mode of security event.	
Available via	IdsM.h	

⌋

8.3.4 IdsM_SetReportingMode

[SWS_IdsM_91053] Definition of API function IdsM_SetReportingMode ⌈

Service Name	IdsM_SetReportingMode	
Syntax	`Std_ReturnType IdsM_SetReportingMode ( IdsM_SecurityEventIdType securityEventId, IdsM_SensorInstanceIdType sensorInstanceId, IdsM_Filters_ReportingModeType reportingMode )`	
Service ID [hex]	0x16	
Sync/Async	Asynchronous	
Reentrancy	Reentrant	
Parameters (in)	securityEventId	External ID of the Security Event from whom the reporting mode shall be set.
	sensorInstanceId	ID of the sensor instance of the security event
	reportingMode	The reporting mode for the requested Security Event
Parameters (inout)	None	
Parameters (out)	None	

▽

△

Return value	Std_ReturnType	E_OK: Request successful E_NOT_OK: Request failed IDSM_E_PARAM_INVALID: Request failed since invalid input argument IDSM_E_BUSY: Request failed since service is busy
Description	This API is the application interface to set the reporting mode of security event.	
Available via	IdsM.h	

⌋

8.3.5 IdsM_GetStatusReportingMode

[SWS_IdsM_91052] Definition of API function IdsM_GetStatusReportingMode ⌈

Service Name	IdsM_GetStatusReportingMode	

Syntax	`Std_ReturnType IdsM_GetStatusReportingMode (`
	`  IdsM_ReportingModeStatusType* status`
	`)`	

Service ID [hex]	0x15	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	None	
Parameters (inout)	None	
Parameters (out)	status	Pointer to where to store the status of the currently running set reporting mode operation. Parameter must not be NULL.
Return value	Std_ReturnType	E_OK: Request successful E_NOT_OK: Request failed
Description	This API is the application interface to get the status of set reporting mode.	
Available via	IdsM.h	

⌋

8.3.6 Callback Notifications

This is a list of functions provided for other modules.

8.3.6.1 IdsM_BswM_StateChanged

[SWS_IdsM_91005] Definition of callback function IdsM_BswM_StateChanged ⌈

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_BswM.h	

⌋

8.3.6.2 IdsM_TpTxConfirmation

[SWS_IdsM_91011] Definition of callback function IdsM_TpTxConfirmation ⌈

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	`E_OK`: The PDU was transmitted successfully. `E_NOT_OK`: Transmission of the PDU failed.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	The function is called to confirm a successful transmission of the TP IdsM-UpperLayerTxPdu or to report an error that occurred during transmission.	
Available via	IdsM_PduR.h	

⌋

8.3.6.3 IdsM_TxConfirmation

[SWS_IdsM_91009] Definition of callback function IdsM_TxConfirmation ⌈

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 successfully. `E_NOT_OK`: Transmission of the PDU failed.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	The function is called to confirm the complete transmission of the IF IdsM-UpperLayerTxPdu.	
Available via	IdsM_PduR.h	

⌋

8.3.6.4 IdsM_CopyTxData

[SWS_IdsM_91010] Definition of callback function IdsM_CopyTxData ⌈

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 request transmit data of the TP IdsM-UpperLayerTxPdu. The function can be called several times and each call to this function copies the next part of the data to be transmitted.	
Available via	IdsM_PduR.h	

⌋

8.3.6.5 IdsM_Dcm_GetReportingMode_RequestResults

[SWS_IdsM_91007] Definition of callback function IdsM_Dcm_GetReporting Mode_RequestResults ⌈

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	DCM_POS_RESP: When Return value is E_OK for successful operation. DCM_E_REQUESTSEQUENCEERROR: When Return value is E_NOT_OK since Routine IdsM_Dcm_GetReportingMode_ RequestResults is requested before Routine IdsM_Dcm_Get ReportingMode_Start is invoked.
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_Dcm.h	

⌋

8.3.6.6 IdsM_Dcm_GetReportingMode_Start

[SWS_IdsM_91006] Definition of callback function IdsM_Dcm_GetReporting Mode_Start ⌈

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	DCM_POS_RESP: When Return value is E_OK for successful operation. DCM_E_REQUESTOUTOFRANGE: When Return value is E_NOT_OK, since the Routine is invoked with following invalid condition: - An invalid reporting mode. - An invalid combination of External Event ID and Sensor Instance ID.
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_Dcm.h	

⌋

8.3.6.7 IdsM_Dcm_SetReportingMode_Start

[SWS_IdsM_91008] Definition of callback function IdsM_Dcm_SetReporting Mode_Start ⌈

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	DCM_POS_RESP: When Return value is E_OK for successful operation. DCM_E_REQUESTOUTOFRANGE: When Return value is E_NOT_OK since the Routine is invoked with following invalid condition: - An invalid reporting mode. - An invalid combination of External Event ID and Sensor Instance ID. DCM_E_BUSYREPEATREQUEST: When Return value is E_NOT_OK, since the Routine is requested to get the status of the previous persistence request for which the status is still Pending (OpStatus = DCM_PENDING). DCM_E_GENERALPROGRAMMINGFAILURE: When Return value is E_NOT_OK, since the Routine is initiated and the Persistence request is unsuccessful. DCM_E_CONDITIONSNOTCORRECT: The Routine is invoked in the following scenarios: 1. Requested for Cancel (OpStatus = DCM_CANCEL) when Persistency to Non-Volatile Memory is not configured. 2. Request for Cancel (OpStatus = DCM_CANCEL) was not successful since writing into Non-Volatile Memory was already started.
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_Dcm.h	

⌋

8.3.6.8 IdsM_DemReadQSEv

[SWS_IdsM_91037] Definition of API function IdsM_DemReadQSEv ⌈

Service Name	IdsM_DemReadQSEv	
Syntax	`Std_ReturnType IdsM_DemReadQSEv ( uint8* DataBuffer, uint16 DataLength, Dcm_NegativeResponseCodeType* ErrorCode )`	
Service ID [hex]	0x12	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	DataLength	size of the buffer pointed by DataBuffer
Parameters (inout)	None	
Parameters (out)	DataBuffer	the buffer into which IdsM shall copy the QSEv into
	ErrorCode	unused (default E_OK)
Return value	Std_ReturnType	`E_OK`: IdsM was able to succesfully copy the QSEv into the requested buffer `E_NOT_OK`: IdsM was unable to copy the QSEv into the requested buffer
Description	IdsM offers this callback function for Dem to fetch the qualified security event (QSEv) to be stored in the freeze frame records.	
Available via	IdsM_Dem.h	

⌋

8.3.6.9 IdsM_CsmNotification

[SWS_IdsM_91034] Definition of callback function IdsM_CsmNotification ⌈

Service Name	IdsM_CsmNotification	
Syntax	`void IdsM_CsmNotification ( uint32 jobId, Crypto_ResultType result )`	
Service ID [hex]	0x10	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	jobId	jobid of the operation that caused the callback
	result	Contains the result of the cryptographic operation.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	This API is used by Csm to notify IdsM that the asynchronous authenticator generation job is complete. The function name is configurable under CsmJob/CsmJobPrimitiveCallback Ref}/CsmCallbackFunc.	
Available via	IdsM_Csm.h	

⌋

8.3.6.10 IdsM_TransmissionSetState

[SWS_IdsM_91036] Definition of API function IdsM_TransmissionSetState ⌈

Service Name	IdsM_TransmissionSetState	
Syntax	`void IdsM_TransmissionSetState ( IdsM_TransmissionStateType state )`	
Service ID [hex]	0x11	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	state	desired transmission state of IdsM
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	This API is used by BswM to enable/disable the forwarding of the qualified security events (QSEvs).	
Available via	IdsM_BswM.h	

⌋

8.4 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.

8.4.1 IdsM_MainFunction

[SWS_IdsM_91000] Definition of scheduled function IdsM_MainFunction ⌈

Service Name	IdsM_MainFunction
Syntax	`void IdsM_MainFunction ( void )`
Service ID [hex]	0x02
Description	This function is called periodically. It processes security events asynchronously which are queued during API function calls.
Available via	IdsM.h

⌋

8.5 Expected Interfaces

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

8.5.1 Mandatory Interfaces

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

[SWS_IdsM_91023] Definition of mandatory interfaces required by module IdsM ⌈

API Function	Header File	Description
There are no mandatory interfaces.		

⌋

8.5.2 Optional Interfaces

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

[SWS_IdsM_91024] Definition of optional interfaces requested by module IdsM ⌈

API Function	Header File	Description
Csm_MacGenerate	Csm.h	Uses the given data to perform a MAC generation and stores the MAC in the memory location pointed to by the MAC pointer.
Csm_SignatureGenerate	Csm.h	Uses the given data to perform the signature calculation and stores the signature in the memory location pointed by the result pointer.
NvM_GetErrorStatus	NvM.h	Service to read the block dependent error/status information.
NvM_WriteBlock	NvM.h	Service to copy the data of the RAM block to its corresponding NV block.
StbM_GetCurrentTime	StbM.h	Returns a time tuple (Local time, Global time and Timebase status) and user data details 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).

⌋

8.5.3 Configurable Interfaces

In this section, all interfaces are listed where the target function could be configured. The target function is usually a callback function. The names of this kind of interfaces are not fixed because they are configurable.

[SWS_IdsM_91033] Definition of configurable interface IdsM_<ContextDataModifierCallout> ⌈

Service Name	IdsM_<ContextDataModifierCallout>	
Syntax	`void IdsM_<ContextDataModifierCallout> ( uint16 securityEventId, const uint8* contextData, uint16 contextDataSize, uint16* modifiedContextDataSize, uint8* modifiedContextData )`	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	securityEventId	Security event ID
	contextData	Pointer to original reported context data. NULL_PTR if no context data is available.
	contextDataSize	Size of original reported context data.
Parameters (inout)	modifiedContextDataSize	in: The maximum size of data that is allowed to be written into the buffer pointed by modifiedContextDataBuffer. This is computed by adding the original reported contextDataSize and the configured ContextDataSizeModifier for the respective modifier callout. out: The actual size of the data that was copied into modified ContextDataBuffer by the application.
Parameters (out)	modifiedContextData	Pointer to the output buffer for the modified context data in big endian format. The application shall set the new context data in this buffer.
Return value	None	
Description	In this callout the application can modify the original context data of a SEv. The application is responsible to copy the resulting context data into the buffer "modifiedContextData". The size of the resulting context data must be returned using pointer "modifiedContextDataSize". The function name of this callout is configurable via [ECUC_IdsM_00067] or [ECUC_IdsM_00070].	
Available via	IdsM_Externals.h	

⌋

Please check chapter 7.3.5 for more information about context data endianess.

8.6 Service Interfaces

8.6.1 Client-Server Interfaces

8.6.1.1 IdsM_IdsMService(EventName)

[SWS_IdsM_91044] Definition of ClientServerInterface IdsMService_{EventName} ⌈

Name	IdsMService_{EventName}		
Comment	Interface to report security events to the IdsM.		
IsService	true		
Variation	({ecuc(IdsM/IdsMConfiguration/IdsMEvent/IdsMExternalEventId)} is in range 0x8000 - 0xFFFE) EventName = {ecuc(IdsM/IdsMConfiguration/IdsMEvent.SHORT-NAME)}		
Possible Errors	–	–	–

Operation	ReportSecurityEvent	
Comment	This function shall report a security event with optional versioned context data, timestamp and count.	
Relates to	IdsM_ReportSecurityEvent	
Variation	–	
Parameters	contextData	
	Type	IdsM_{EventName}_ContextDataType
	Direction	IN
	Comment	Pointer to optional context data descriptor in big endian format.
	Variation	({ecuc(IdsM/IdsMConfiguration/IdsMEvent/IdsMServiceInterfaceOptions/IdsMEventMaxContextDataSize)} > 0
	contextDataSize	
	Type	uint16
	Direction	IN
	Comment	Size of context data. If no context data is available, provide 0.
	Variation	({ecuc(IdsM/IdsMConfiguration/IdsMEvent/IdsMServiceInterfaceOptions/IdsMEventMaxContextDataSize)} > 0
	contextDataVersion	
	Type	uint16
	Direction	IN
	Comment	Version of context data. If no context data is available, provide 1.
	Variation	({ecuc(IdsM/IdsMConfiguration/IdsMEvent/IdsMServiceInterfaceOptions/IdsMEventMaxContextDataSize)} > 0
	count	
	Type	uint16
	Direction	IN
	Comment	Number of occurrences of the security events since last report. Permitted range [1, max(uint16)]. If sensor didn't maintain a special count, provide 1.
	Variation	–
	timestamp	
	Type	IdsM_TimestampDataType
	Direction	IN
	Comment	Pointer to timestamp used for time reference of the security event.
	Variation	({ecuc(IdsM/IdsMConfiguration/IdsMEvent/IdsMServiceInterfaceOptions/IdsMEventEnableTimestampReporting)} == True
Possible Errors	–	

⌋

The C/S interface requires wrapper functions at the BSW C level to match the arguments between C/S interfaces and the mapped BSW API (related to API). Please also check chapter 7.3.5 for more information about context data endianess.

8.6.1.2 IdsM_RequestCustomTimestamp

[SWS_IdsM_91042] Definition of ClientServerInterface IdsM_RequestCustomTimestamp ⌈

Name	IdsM_RequestCustomTimestamp		
Comment	Interface to request custom timestamps from the application.		
IsService	true		
Variation	{ecuc(IdsM/IdsMGeneral/IdsMTimestamp/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.	
Relates to	–	
Variation	–	
Parameters	timestamp	
	Type	IdsM_TimestampDataType
	Direction	OUT
	Comment	Timestamp requested by the IdsM from a custom time source.
	Variation	–
Possible Errors	E_OK E_NOT_OK	

⌋

8.6.1.3 IdsM_ReportingModeService

[SWS_IdsM_91054] Definition of ClientServerInterface IdsMReportingModeService ⌈

Name	IdsMReportingModeService		
Comment	Interface to set and get the reporting mode of security events		
IsService	true		
Variation	{ecuc(IdsM/IdsMGeneral/IdsMReportingModeApi)} == True		
Possible Errors	0	E_OK	Operation successful
	1	E_NOT_OK	Operation failed
	0x03	IDSM_E_BUSY	–
	0x0A	IDSM_E_PARAM_INVALID	–

Operation	GetReportingMode	
Comment	–	
Relates to	IdsM_GetReportingMode	
Variation	–	
Parameters	securityEventId	
	Type	IdsM_SecurityEventIdType
	Direction	IN
	Comment	External ID of the Security Event from whom the reporting mode shall be returned
	Variation	–

▽

△

	sensorInstanceId	
	Type	IdsM_SensorInstanceIdType
	Direction	IN
	Comment	ID of the sensor instance of the security event
	Variation	–
	reportingMode	
	Type	IdsM_Filters_ReportingModeType
	Direction	OUT
	Comment	The reporting mode for the requested Security Event
	Variation	–
Possible Errors	E_OK E_NOT_OK IDSM_E_PARAM_INVALID	

Operation	GetStatusReportingMode	
Comment	–	
Relates to	IdsM_GetStatusReportingMode	
Variation	–	
Parameters	status	
	Type	IdsM_ReportingModeStatusType
	Direction	OUT
	Comment	–
	Variation	–
Possible Errors	E_OK E_NOT_OK	

Operation	SetReportingMode	
Comment	–	
Relates to	IdsM_SetReportingMode	
Variation	–	
Parameters	securityEventId	
	Type	IdsM_SecurityEventIdType
	Direction	IN
	Comment	External ID of the Security Event from whom the reporting mode shall be set.
	Variation	–
	sensorInstanceId	
	Type	IdsM_SensorInstanceIdType
	Direction	IN
	Comment	ID of the sensor instance of the security event
	Variation	–
	reportingMode	
	Type	IdsM_Filters_ReportingModeType
	Direction	IN
	Comment	The reporting mode for the requested Security Event
	Variation	–

▽

△

Possible Errors	E_OK E_NOT_OK IDSM_E_BUSY IDSM_E_PARAM_INVALID

⌋

8.6.2 Implementation Data Types

[SWS_IdsM_91016] Definition of ImplementationDataType IdsM_{EventName}_ContextDataType ⌈

Name	IdsM_{EventName}_ContextDataType		
Kind	Array	***Element type***	uint8
Size	{ecuc(IdsM/IdsMConfiguration/IdsMEvent/IdsMServiceInterfaceOptions/IdsMEventMaxContextDataSize)} Elements		
Description	Data type for IdsM context data.		
Variation	EventName = {ecuc(IdsM/IdsMConfiguration/IdsMEvent.SHORT-NAME)}		
Available via	Rte_IdsM_Type.h		

⌋

[SWS_IdsM_91039] Definition of ImplementationDataType IdsM_TimestampDataType ⌈

Name	IdsM_TimestampDataType		
Kind	Array	***Element type***	uint8
Size	8 Elements		
Description	Data type for IdsM timestamps.		
Variation	–		
Available via	Rte_IdsM_Type.h		

⌋

8.6.3 Ports

8.6.3.1 Port IdsM_IdsMService

[SWS_IdsM_91030] Definition of Port IdsMService_{EventName} provided by module IdsM ⌈

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 - 0xFFFE) ({ecuc(IdsM/IdsMConfiguration/IdsMEvent/IdsMServiceInterfaceOptions}} EXISTS EventName = {ecuc(IdsM/IdsMConfiguration/IdsMEvent.SHORT-NAME)}	

⌋

8.6.3.2 Port IdsM_RequestCustomTimestamp

[SWS_IdsM_91043] Definition of Port IdsM_RequestCustomTimestamp required by module IdsM ⌈

Name	IdsM_RequestCustomTimestamp		
Kind	RequiredPort	***Interface***	IdsM_RequestCustomTimestamp
Description	–		
Variation	–		

⌋

8.6.3.3 Port IdsM_ReportingModeService

[SWS_IdsM_91055] Definition of Port IdsM_ReportingModeStatus provided by module IdsM ⌈

Name	IdsM_ReportingModeStatus		
Kind	ProvidedPort	***Interface***	IdsMReportingModeService
Description	–		
Variation	ecuc(IdsM/IdsMGeneral/IdsMReportingModeApi)} == True		

⌋

8.6.4 Callout Interfaces

8.6.4.1 IdsMContextDataModifierCallout

[ECUC_IdsM_00065] Definition of EcucParamConfContainerDef IdsMContext DataModifierCallout ⌈

Container Name	IdsMContextDataModifierCallout
Parent Container	IdsMContextDataModification
Upstream mapping defined	–
Description	Configuration of event specific context data modifier callouts.

▽

△

Multiplicity	0..65535
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
IdsMContextDataModifierCalloutFct	1	[ECUC_IdsM_00067]
IdsMContextDataSizeModifier	1	[ECUC_IdsM_00066]

No Included Containers

⌋

[ECUC_IdsM_00067] Definition of EcucFunctionNameDef IdsMContextDataModifierCalloutFct ⌈

Parameter Name	IdsMContextDataModifierCalloutFct
Parent Container	IdsMContextDataModifierCallout
Upstream mapping defined	–
Description	Event specific context data modifier callout function. If the global context data modifier callout is configured, it can be overruled by this event-specific callout.
Multiplicity	1
Type	EcucFunctionNameDef
Default value	–
Regular Expression	–
Pre-Compile only	true

⌋

[ECUC_IdsM_00066] Definition of EcucIntegerParamDef IdsMContextDataSizeModifier ⌈

Parameter Name	IdsMContextDataSizeModifier	
Parent Container	IdsMContextDataModifierCallout	
Upstream mapping defined	–	
Description	Event specific context data size modifier used in combination with event specific context data size modifier callout function.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	-1499 .. 1499	
Default value	0	
Pre-Compile only	true	

⌋

9 Sequence diagrams

9.1 Sequence diagram for storage of qualified security events in Dem

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

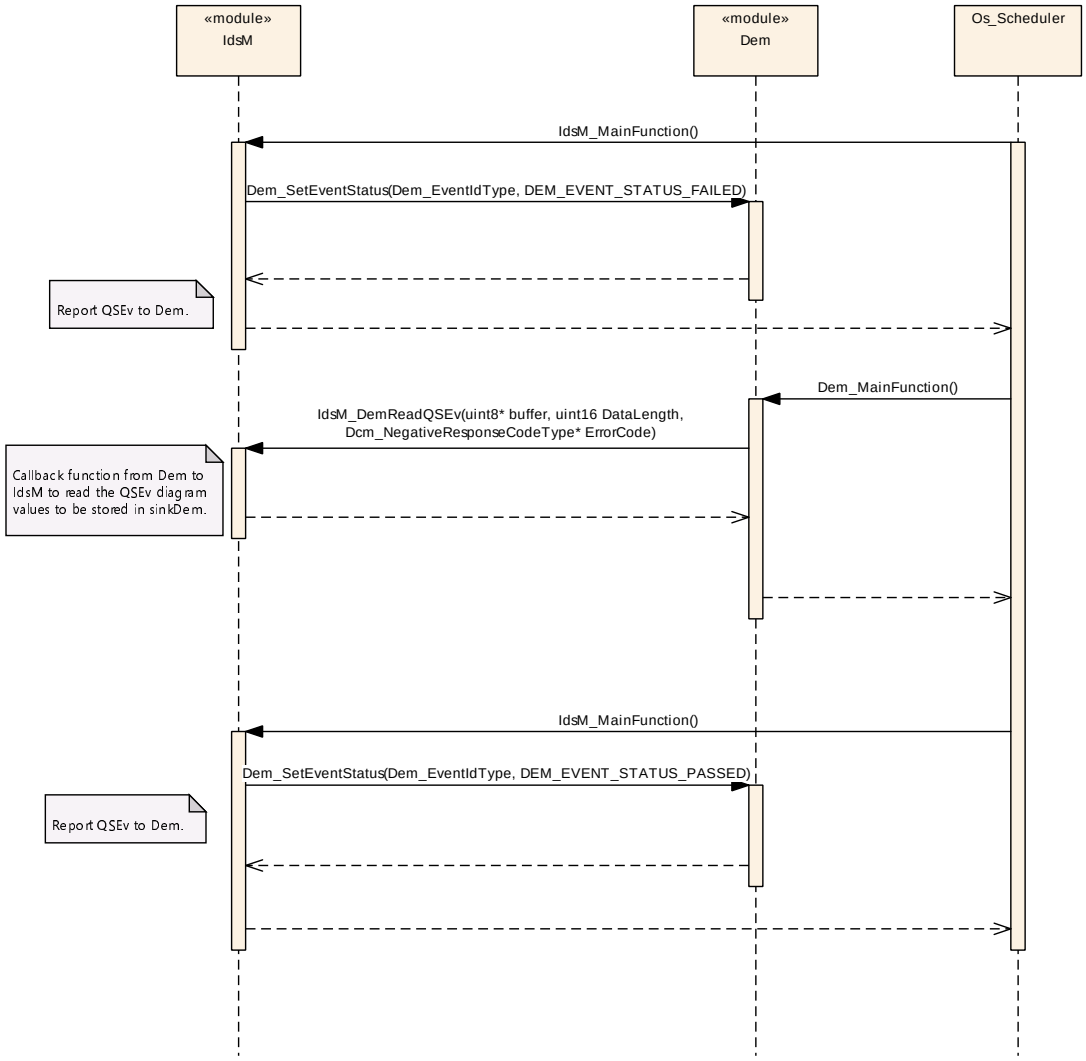


Figure 9.1: Dem sink for Single / Multipartition use case

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.

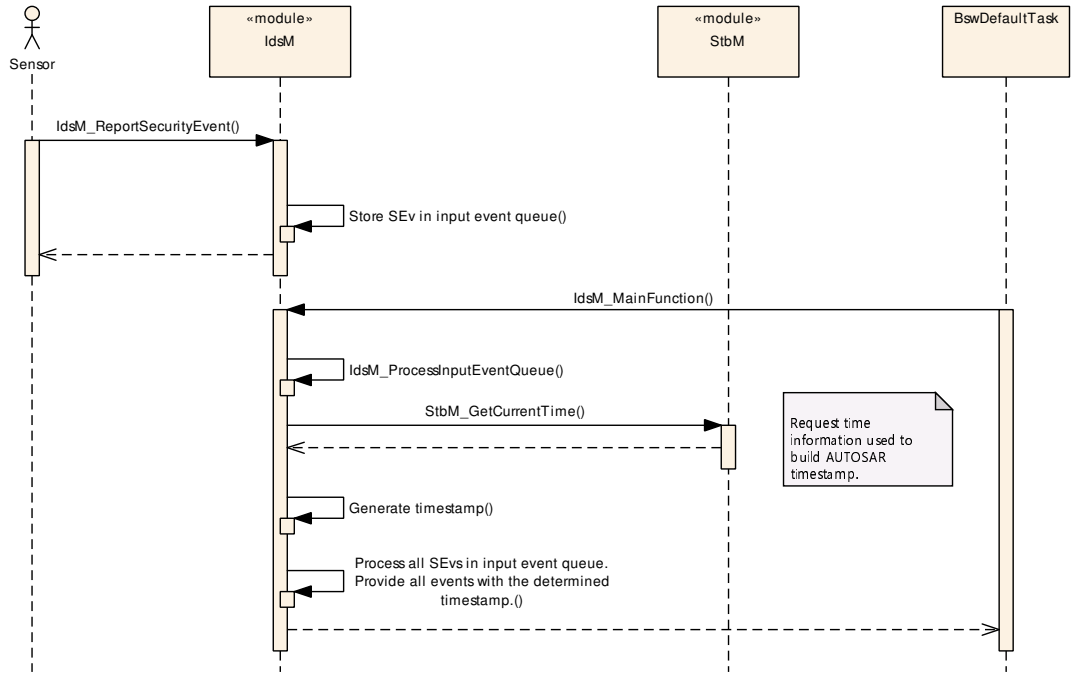


Figure 9.2: AUTOSAR Timestamp: The StbM is used as source for timestamp data

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.

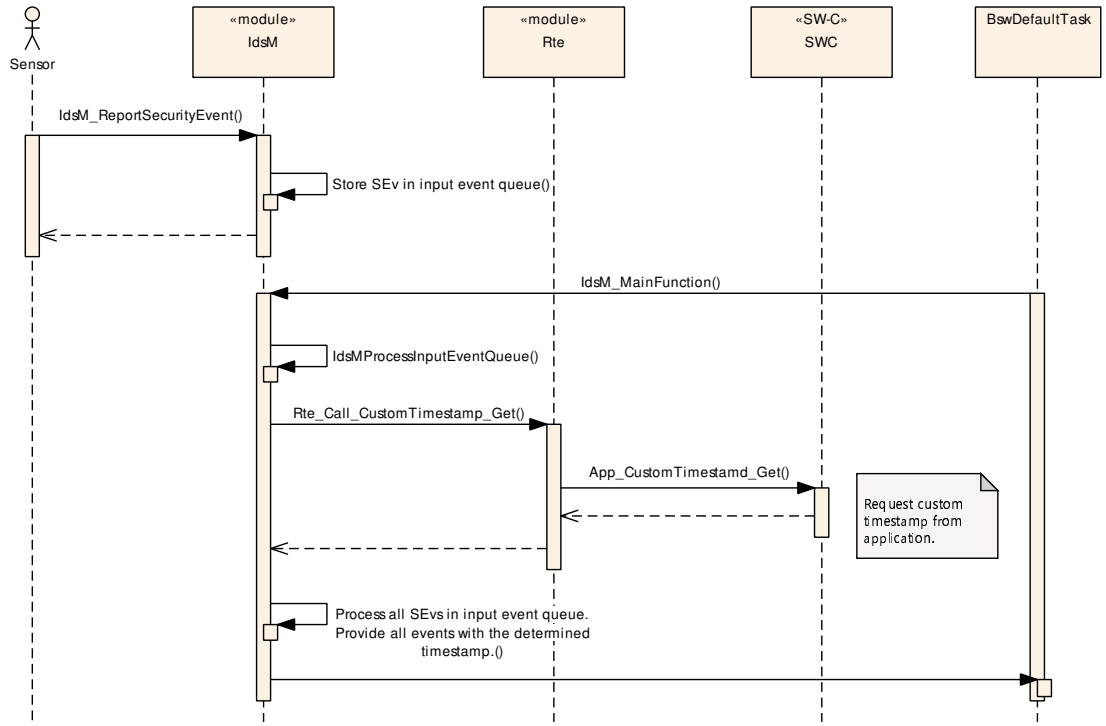


Figure 9.3: Custom Timestamp: Timestamps are requested from the application

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

[ECUC_IdsM_00001] Definition of EcucModuleDef IdsM ⌈

Module Name	IdsM
Description	Configuration of the IdsM module.
Post-Build Selectable Support	true
Post-Build Loadable Support	false

Included Containers		
Container Name	**Multiplicity**	**Description**
IdsMConfiguration	1	Configuration parameters of the module IdsM.
IdsMGeneral	1	General configuration parameters of IdsM.

⌋

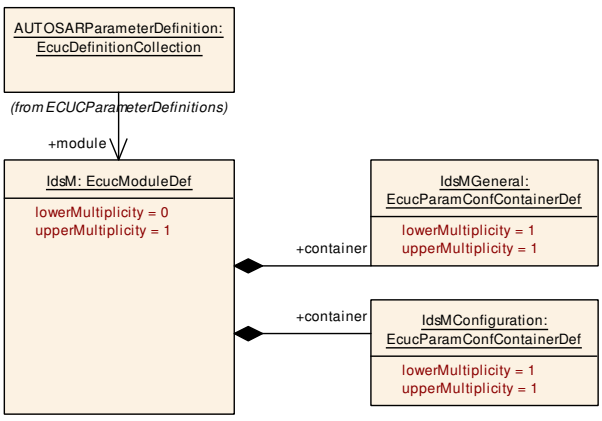


Figure 10.1: Intrusion Detection System Manager Overview

10.1.2 IdsMGeneral

[ECUC_IdsM_00002] Definition of EcucParamConfContainerDef IdsMGeneral ⌈

Container Name	IdsMGeneral
Parent Container	IdsM
Upstream mapping defined	–
Description	General configuration parameters of IdsM.
Multiplicity	1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
IdsMAuthenticatorSupport	1	[ECUC_IdsM_00009]
IdsMDevErrorDetect	1	[ECUC_IdsM_00005]
IdsMDiagnosticSupport	1	[ECUC_IdsM_00010]
IdsMEnableSecurityEventReporting	1	[ECUC_IdsM_00085]
IdsMInstanceId	1	[ECUC_IdsM_00007]
IdsMMainFunctionPeriod	1	[ECUC_IdsM_00004]
IdsMReportingModeApi	1	[ECUC_IdsM_00092]
IdsMTimestampSupport	1	[ECUC_IdsM_00084]
IdsMNvmBlockDescriptor	0..1	[ECUC_IdsM_00013]

Included Containers		
Container Name	**Multiplicity**	**Description**
IdsMAuthenticator	0..1	If this container exists all qualified security events are authenticated by the crypto service.
IdsMGlobalRateLimitationFilters	0..1	Global rate limitation filters for all SEvs.
IdsMSecurityEventRefs	0..1	Container for the references to IdsMEvent elements representing the security events that the IdsM module shall report to itself in case the corresponding security related event occurs.
IdsMTimestamp	0..1	If this container exists a timestamp field is added to all qualified security events.

⌋

[ECUC_IdsM_00009] Definition of EcucBooleanParamDef IdsMAuthenticatorSupport ⌈

Parameter Name	IdsMAuthenticatorSupport
Parent Container	IdsMGeneral
Upstream mapping defined	[5, FO_TPS_SecurityExtractTemplate]
Description	This parameter enables/disables the functionality of sending messages to the network with an authenticator calculated by the crypto services.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

⌋

[ECUC_IdsM_00005] Definition of EcucBooleanParamDef IdsMDevErrorDetect ⌈

Parameter Name	IdsMDevErrorDetect
Parent Container	IdsMGeneral
Upstream mapping defined	–
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.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_IdsM_00010] Definition of EcucBooleanParamDef IdsMDiagnosticSupport ⌈

Parameter Name	IdsMDiagnosticSupport
Parent Container	IdsMGeneral
Upstream mapping defined	–
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
Multiplicity	1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

⌋

[ECUC_IdsM_00085] Definition of EcucBooleanParamDef IdsMEnableSecurityEventReporting ⌈

Parameter Name	IdsMEnableSecurityEventReporting
Parent Container	IdsMGeneral
Upstream mapping defined	–
Description	Switches the reporting of internal security events. • true: reporting is enabled. • false: reporting is disabled.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

⌋

[ECUC_IdsM_00007] Definition of EcucIntegerParamDef IdsMInstanceId ⌈

Parameter Name	IdsMInstanceId	
Parent Container	IdsMGeneral	
Upstream mapping defined	[5, FO_TPS_SecurityExtractTemplate]	
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.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 1023	
Default value	–	
Post-Build Multiplicity	-	Selectable
Post-Build Value	-	Selectable

⌋

[ECUC_IdsM_00004] Definition of EcucFloatParamDef IdsMMainFunctionPeriod ⌈

Parameter Name	IdsMMainFunctionPeriod	
Parent Container	IdsMGeneral	
Upstream mapping defined	–	
Description	The period between successive calls to the IdsM main function (as float in seconds).	
Multiplicity	1	
Type	EcucFloatParamDef	
Range	[0 .. INF]	
Default value	0.01	
Pre-Compile only	true	

⌋

[ECUC_IdsM_00092] Definition of EcucBooleanParamDef IdsMReportingModeApi ⌈

Parameter Name	IdsMReportingModeApi
Parent Container	IdsMGeneral
Upstream mapping defined	–
Description	This parameter enables/disables the functions IdsM_GetReportingMode, IdsM_SetReportingMode and IdsM_GetStatusReportingMode and the service IdsMReportingModeService to set and get the Reporting Mode, minor and patch version information of the module.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

⌋

[ECUC_IdsM_00084] Definition of EcucBooleanParamDef IdsMTimestampSupport ⌈

Parameter Name	IdsMTimestampSupport
Parent Container	IdsMGeneral
Upstream mapping defined	–
Description	This parameter enables/disables the functionality of timestamping and accepting timestamps for reported SEvs.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

⌋

[ECUC_IdsM_00013] Definition of EcucReferenceDef IdsMNvmBlockDescriptor ⌈

Parameter Name	IdsMNvmBlockDescriptor
Parent Container	IdsMGeneral
Upstream mapping defined	–
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
Multiplicity	0..1
Type	Reference to NvMBlockDescriptor
Pre-Compile only	true

⌋

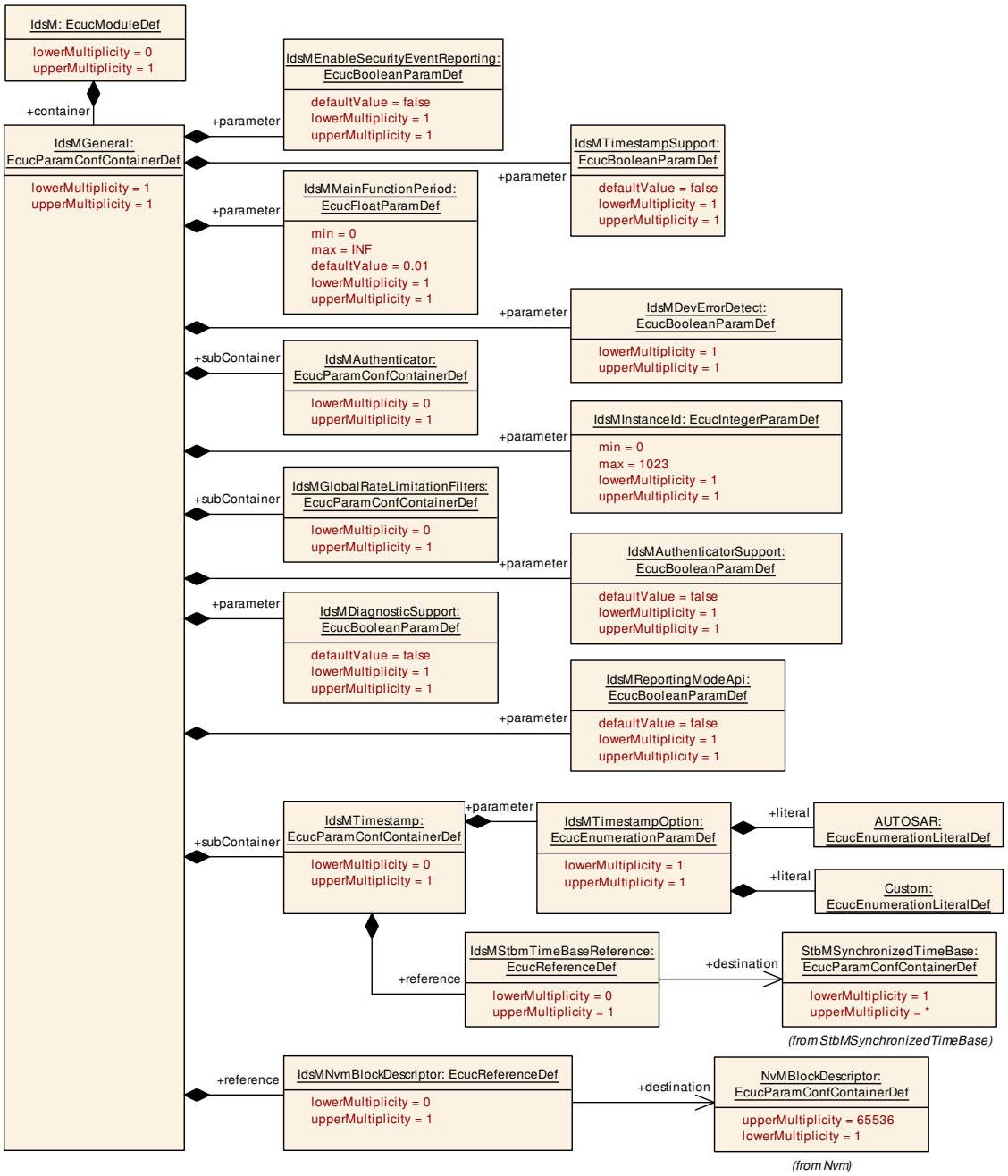


Figure 10.2: IdsM general configuration overview

10.1.3 IdsMGlobalRateLimitationFilter

[ECUC_IdsM_00008] Definition of EcucParamConfContainerDef IdsMGlobalRateLimitationFilters ⌈

Container Name	IdsMGlobalRateLimitationFilters
Parent Container	IdsMGeneral
Upstream mapping defined	[5, FO_TPS_SecurityExtractTemplate]
Description	Global rate limitation filters for all SEvs.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

No Included Parameters

Included Containers		
Container Name	Multiplicity	Description
IdsMFilterEventRateLimitation	0..1	For configurable time intervals of length "IdsMRateLimitation TimeInterval" 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.
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 "IdsMTrafficLimitationMaximum Bytes". 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.

⌋

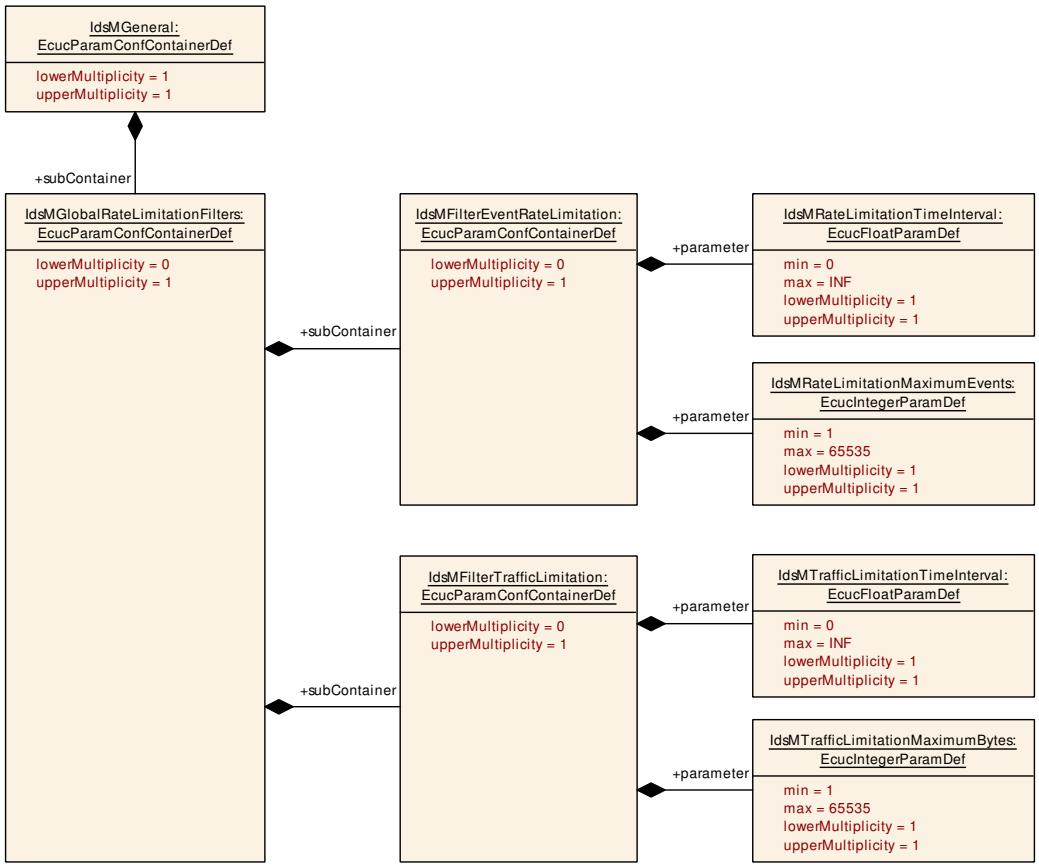


Figure 10.3: IdsM global rate limitation overview

10.1.4 IdsMSecurityEventRefs

[ECUC_IdsM_00073] Definition of EcucParamConfContainerDef IdsMSecurity EventRefs ⌈

Container Name	IdsMSecurityEventRefs
Parent Container	IdsMGeneral
Upstream mapping defined	–
Description	Container for the references to IdsMEvent elements representing the security events that the IdsM module shall report to itself in case the corresponding security related event occurs.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
SEV_IDSM_COMMUNICATION_ERROR	0..1	[ECUC_IdsM_00077]
SEV_IDSM_NO_CONTEXT_DATA_BUFFER_AVAILABLE	0..1	[ECUC_IdsM_00075]
SEV_IDSM_NO_EVENT_BUFFER_AVAILABLE	0..1	[ECUC_IdsM_00074]
SEV_IDSM_NO_QUALIFIED_EVENT_BUFFER_-AVAILABLE	0..1	[ECUC_IdsM_00081]

No Included Containers

⌋

[ECUC_IdsM_00077] Definition of EcucReferenceDef SEV_IDSM_COMMUNICATION_ERROR ⌈

Parameter Name	SEV_IDSM_COMMUNICATION_ERROR
Parent Container	IdsMSecurityEventRefs
Upstream mapping defined	–
Description	An error occurred when sending a QSEv via PDU.
Multiplicity	0..1
Type	Symbolic name reference to IdsMEvent
Pre-Compile only	true

⌋

[ECUC_IdsM_00075] Definition of EcucReferenceDef SEV_IDSM_NO_CONTEXT_DATA_BUFFER_AVAILABLE ⌈

Parameter Name	SEV_IDSM_NO_CONTEXT_DATA_BUFFER_AVAILABLE
Parent Container	IdsMSecurityEventRefs
Upstream mapping defined	–
Description	The context data of an incoming event cannot be stored because there are no more context data buffers available.
Multiplicity	0..1
Type	Symbolic name reference to IdsMEvent
Pre-Compile only	true

⌋

[ECUC_IdsM_00074] Definition of EcucReferenceDef SEV_IDSM_NO_EVENT_BUFFER_AVAILABLE ⌈

Parameter Name	SEV_IDSM_NO_EVENT_BUFFER_AVAILABLE
Parent Container	IdsMSecurityEventRefs
Upstream mapping defined	–
Description	A SEv cannot be handled because there are no more event buffers available to process the event.
Multiplicity	0..1
Type	Symbolic name reference to IdsMEvent
Pre-Compile only	true

⌋

[ECUC_IdsM_00081] Definition of EcucReferenceDef SEV_IDSM_NO_QUALIFIED_EVENT_BUFFER_AVAILABLE ⌈

Parameter Name	SEV_IDSM_NO_QUALIFIED_EVENT_BUFFER_AVAILABLE
Parent Container	IdsMSecurityEventRefs
Upstream mapping defined	–
Description	A security event is raised when a QSEv has to be dropped due to insufficient QSEv buffers available.
Multiplicity	0..1
Type	Symbolic name reference to IdsMEvent
Pre-Compile only	true

⌋

10.1.5 IdsMFilterEventRateLimitation

[ECUC_IdsM_00053] Definition of EcucParamConfContainerDef IdsMFilterEventRateLimitation ⌈

Container Name	IdsMFilterEventRateLimitation
Parent Container	IdsMGlobalRateLimitationFilters
Upstream mapping defined	[5, FO_TPS_SecurityExtractTemplate]
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.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
IdsMRateLimitationMaximumEvents	1	[ECUC_IdsM_00055]
IdsMRateLimitationTimeInterval	1	[ECUC_IdsM_00054]

No Included Containers

⌋

[ECUC_IdsM_00055] Definition of EcucIntegerParamDef IdsMRateLimitationMaximumEvents ⌈

Parameter Name	IdsMRateLimitationMaximumEvents	
Parent Container	IdsMFilterEventRateLimitation	
Upstream mapping defined	[5, FO_TPS_SecurityExtractTemplate]	
Description	The maximum number of SEvs which are passed on by this filter in a single rate limitation time interval.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	1 .. 65535	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_IdsM_00054] Definition of EcucFloatParamDef IdsMRateLimitationTimeInterval ⌈

Parameter Name	IdsMRateLimitationTimeInterval	
Parent Container	IdsMFilterEventRateLimitation	
Upstream mapping defined	[5, FO_TPS_SecurityExtractTemplate]	
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.	
Multiplicity	1	
Type	EcucFloatParamDef	
Range	[0 .. INF]	
Default value	–	
Pre-Compile only	true	

⌋

10.1.6 IdsMFilterTrafficLimitation

[ECUC_IdsM_00056] Definition of EcucParamConfContainerDef IdsMFilterTrafficLimitation ⌈

Container Name	IdsMFilterTrafficLimitation
Parent Container	IdsMGlobalRateLimitationFilters
Upstream mapping defined	[5, FO_TPS_SecurityExtractTemplate]

▽

△

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.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
IdsMTrafficLimitationMaximumBytes	1	[ECUC_IdsM_00058]
IdsMTrafficLimitationTimeInterval	1	[ECUC_IdsM_00057]

No Included Containers

⌋

[ECUC_IdsM_00058] Definition of EcucIntegerParamDef IdsMTrafficLimitationMaximumBytes ⌈

Parameter Name	IdsMTrafficLimitationMaximumBytes	
Parent Container	IdsMFilterTrafficLimitation	
Upstream mapping defined	[5, FO_TPS_SecurityExtractTemplate]	
Description	The maximum number of bytes to be sent out by the IdsM in a single traffic limitation time interval.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	1 .. 65535	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_IdsM_00057] Definition of EcucFloatParamDef IdsMTrafficLimitationTimeInterval ⌈

Parameter Name	IdsMTrafficLimitationTimeInterval
Parent Container	IdsMFilterTrafficLimitation
Upstream mapping defined	[5, FO_TPS_SecurityExtractTemplate]
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.
Multiplicity	1
Type	EcucFloatParamDef

▽

△

Range	[0 .. INF]	
Default value	–	
Pre-Compile only	true	

⌋

10.1.7 IdsMTimestamp

[ECUC_IdsM_00060] Definition of EcucParamConfContainerDef IdsMTimestamp ⌈

Container Name	IdsMTimestamp
Parent Container	IdsMGeneral
Upstream mapping defined	[5, FO_TPS_SecurityExtractTemplate]
Description	If this container exists a timestamp field is added to all qualified security events.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
IdsMTimestampOption	1	[ECUC_IdsM_00012]
IdsMStbmTimeBaseReference	0..1	[ECUC_IdsM_00014]

No Included Containers

⌋

[ECUC_IdsM_00012] Definition of EcucEnumerationParamDef IdsMTimestamp Option ⌈

Parameter Name	IdsMTimestampOption	
Parent Container	IdsMTimestamp	
Upstream mapping defined	[5, FO_TPS_SecurityExtractTemplate]	
Description	This parameter speciies if the origin of the timestamp is from the AUTOSAR stack or from the application (custom timestamp).	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	AUTOSAR	Timestamp from AUTOSAR stack
	Custom	Timestamp from application
Pre-Compile only	true	

⌋

[ECUC_IdsM_00014] Definition of EcucReferenceDef IdsMStbmTimeBaseReference ⌈

Parameter Name	IdsMStbmTimeBaseReference
Parent Container	IdsMTimestamp
Upstream mapping defined	–
Description	This parameter references the time source when the origin of the timestamp is AUTOSAR.
Multiplicity	0..1
Type	Reference to StbMSynchronizedTimeBase
Pre-Compile only	true

⌋

10.1.8 IdsMAuthenticator

[ECUC_IdsM_00059] Definition of EcucParamConfContainerDef IdsMAuthenticator ⌈

Container Name	IdsMAuthenticator
Parent Container	IdsMGeneral
Upstream mapping defined	[5, FO_TPS_SecurityExtractTemplate]
Description	If this container exists all qualified security events are authenticated by the crypto service.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
IdsMAuthenticatorLength	1	[ECUC_IdsM_00011]
IdsMCsmJobReference	1	[ECUC_IdsM_00015]

No Included Containers

⌋

[ECUC_IdsM_00011] Definition of EcucIntegerParamDef IdsMAuthenticatorLength ⌈

Parameter Name	IdsMAuthenticatorLength
Parent Container	IdsMAuthenticator
Upstream mapping defined	[5, FO_TPS_SecurityExtractTemplate]
Description	This parameter defines the length of the authenticator in bytes calculated by the crypto service. Dependency: SWS_IdsM_CONSTR_00006
Multiplicity	1
Type	EcucIntegerParamDef

▽

△

Range	1 .. 65535	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_IdsM_00015] Definition of EcucReferenceDef IdsMCsmJobReference ⌈

Parameter Name	IdsMCsmJobReference
Parent Container	IdsMAuthenticator
Upstream mapping defined	–
Description	This parameter references the Csm job that is used to generate authenticator when qualified security events must be authenticated.
Multiplicity	1
Type	Reference to CsmJob
Pre-Compile only	true

⌋

10.1.9 IdsMConfiguration

[ECUC_IdsM_00003] Definition of EcucParamConfContainerDef IdsMConfiguration ⌈

Container Name	IdsMConfiguration
Parent Container	IdsM
Upstream mapping defined	–
Description	Configuration parameters of the module IdsM.
Multiplicity	1
Pre-Compile only	true
Configuration Parameters	

No Included Parameters

Included Containers		
Container Name	**Multiplicity**	**Description**
IdsMBlockState	0..16	Configuration of an IdsM blocking state used in the IdsMState BlockFilter to suspend the collection of security events. The active state is reported by the BswM via IdsM_BswM_State Changed().
IdsMBufferConfiguration	1	Configuration of the event buffers and context data buffers used by IdsM.
IdsMContextDataModification	0..1	Configuration of context data modifier callouts.
IdsMEvent	1..65535	Configuration of the IdsM Event unit which is reported by a sensor and its parameters.

▽

△

Included Containers		
Container Name	**Multiplicity**	**Description**
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)
IdsMPdus	0..1	Configuration of the PDU references used to send the events data.
IdsMStatistics	0..1	Configuration parameters for statistics.

⌋

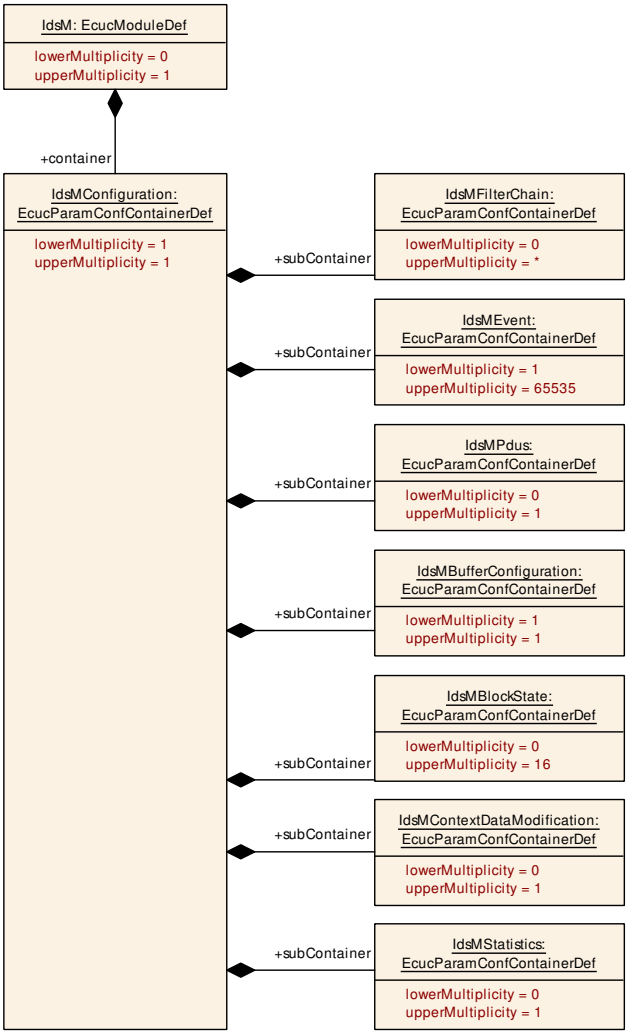


Figure 10.4: IdsM configuration overview

10.1.10 IdsMFilterChain

[ECUC_IdsM_00016] Definition of EcucParamConfContainerDef IdsMFilterChain
⌈

Container Name	IdsMFilterChain
Parent Container	IdsMConfiguration
Upstream mapping defined	[5, FO_TPS_SecurityExtractTemplate]
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)
Multiplicity	0..*
Pre-Compile only	true
Configuration Parameters	

No Included Parameters

Included Containers		
Container Name	**Multiplicity**	**Description**
IdsMBlockStateFilter	0..1	This state filter drops SEvs if the current State reported by the BswM is in this state filter list.
IdsMEventAggregationFilter	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 "Ids MContextDataSourceSelector". 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.
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.
IdsMForwardEveryNthFilter	0..1	Out of all incoming SEVs, drop all but every nth. Those will be forwarded without modification.

⌋

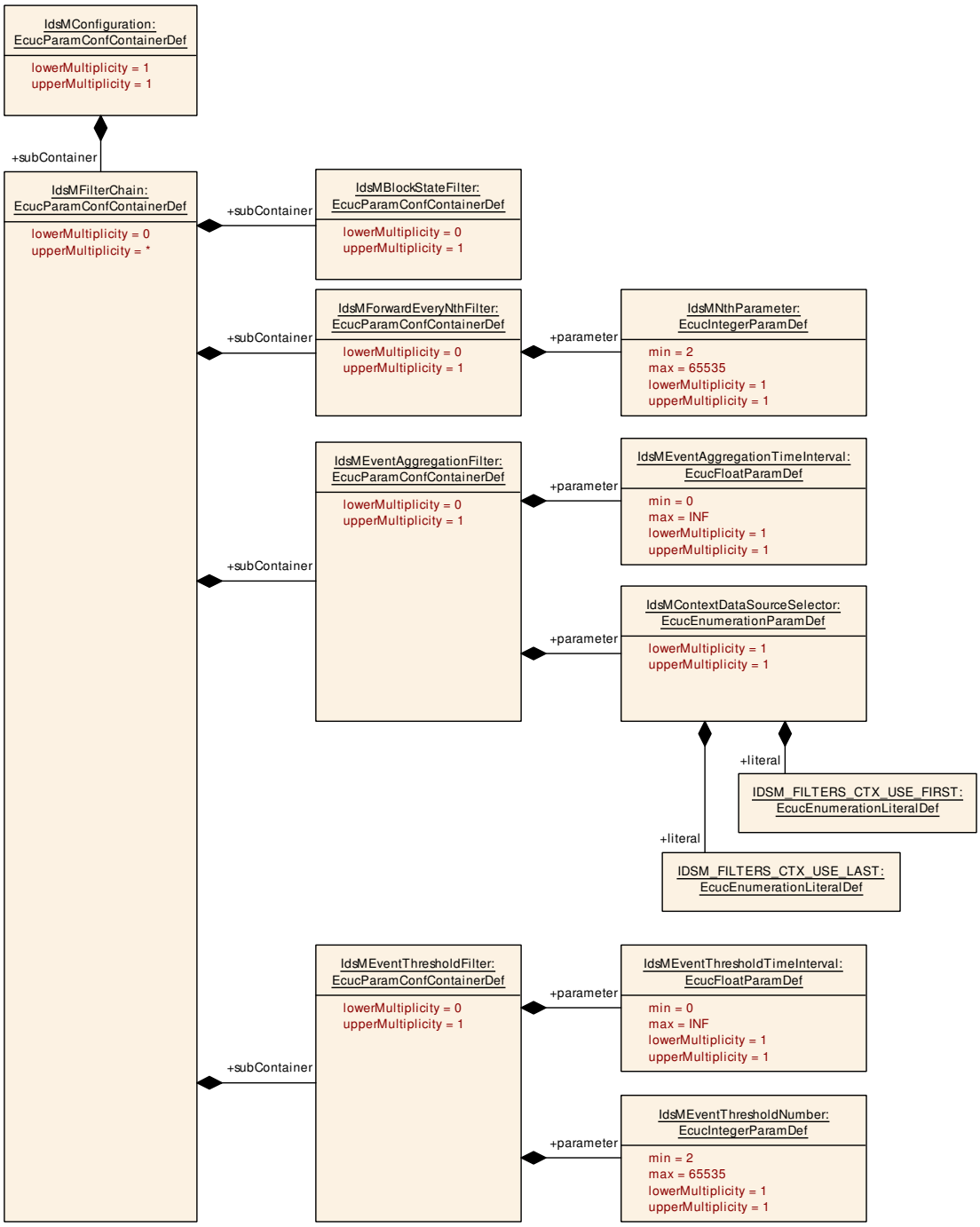


Figure 10.5: IdsM filter chain overview

10.1.11 IdsMBlockStateFilter

[ECUC_IdsM_00021] Definition of EcucParamConfContainerDef IdsMBlockState Filter ⌈

Container Name	IdsMBlockStateFilter
Parent Container	IdsMFilterChain
Upstream mapping defined	[5, FO_TPS_SecurityExtractTemplate]
Description	This state filter drops SEvs if the current State reported by the BswM is in this state filter list.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
IdsMBlockStateReference	1..16	[ECUC_IdsM_00051]

No Included Containers

⌋

[ECUC_IdsM_00051] Definition of EcucReferenceDef IdsMBlockStateReference ⌈

Parameter Name	IdsMBlockStateReference
Parent Container	IdsMBlockStateFilter
Upstream mapping defined	[5, FO_TPS_SecurityExtractTemplate]
Description	The collection of SEvs during this state will be suspended.
Multiplicity	1..16
Type	Reference to IdsMBlockState
Pre-Compile only	true

⌋

10.1.12 IdsMForwardEveryNthFilter

[ECUC_IdsM_00022] Definition of EcucParamConfContainerDef IdsMForwardEveryNthFilter ⌈

Container Name	IdsMForwardEveryNthFilter
Parent Container	IdsMFilterChain
Upstream mapping defined	[5, FO_TPS_SecurityExtractTemplate]
Description	Out of all incoming SEVs, drop all but every nth. Those will be forwarded without modification.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
IdsMNthParameter	1	[ECUC_IdsM_00023]

No Included Containers

⌋

[ECUC_IdsM_00023] Definition of EcucIntegerParamDef IdsMNthParameter ⌈

Parameter Name	IdsMNthParameter	
Parent Container	IdsMForwardEveryNthFilter	
Upstream mapping defined	[5, FO_TPS_SecurityExtractTemplate]	
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.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	2 .. 65535	
Default value	–	
Pre-Compile only	true	

⌋

10.1.13 IdsMEventAggregationFilter

[ECUC_IdsM_00024] Definition of EcucParamConfContainerDef IdsMEventAggregationFilter ⌈

Container Name	IdsMEventAggregationFilter
Parent Container	IdsMFilterChain
Upstream mapping defined	[5, FO_TPS_SecurityExtractTemplate]
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.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
IdsMContextDataSourceSelector	1	[ECUC_IdsM_00026]
IdsMEventAggregationTimeInterval	1	[ECUC_IdsM_00025]

No Included Containers

⌋

[ECUC_IdsM_00026] Definition of EcucEnumerationParamDef IdsMContextDataSourceSelector ⌈

Parameter Name	IdsMContextDataSourceSelector	
Parent Container	IdsMEventAggregationFilter	
Upstream mapping defined	[5, FO_TPS_SecurityExtractTemplate]	
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.	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	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.
Pre-Compile only	true	

⌋

[ECUC_IdsM_00025] Definition of EcucFloatParamDef IdsMEventAggregationTimeInterval ⌈

Parameter Name	IdsMEventAggregationTimeInterval	
Parent Container	IdsMEventAggregationFilter	
Upstream mapping defined	[5, FO_TPS_SecurityExtractTemplate]	
Description	Length of the aggregation time interval (as float in seconds). Note: Shall be configured as a multiple of the IdsM main function period.	
Multiplicity	1	
Type	EcucFloatParamDef	
Range	[0 .. INF]	
Default value	–	
Pre-Compile only	true	

⌋

10.1.14 IdsMEventThresholdFilter

[ECUC_IdsM_00027] Definition of EcucParamConfContainerDef IdsMEventThresholdFilter ⌈

Container Name	IdsMEventThresholdFilter
Parent Container	IdsMFilterChain
Upstream mapping defined	[5, FO_TPS_SecurityExtractTemplate]

▽

△

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.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
IdsMEventThresholdNumber	1	[ECUC_IdsM_00029]
IdsMEventThresholdTimeInterval	1	[ECUC_IdsM_00028]

No Included Containers

⌋

[ECUC_IdsM_00029] Definition of EcucIntegerParamDef IdsMEventThreshold Number ⌈

Parameter Name	IdsMEventThresholdNumber	
Parent Container	IdsMEventThresholdFilter	
Upstream mapping defined	[5, FO_TPS_SecurityExtractTemplate]	
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.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	2 .. 65535	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_IdsM_00028] Definition of EcucFloatParamDef IdsMEventThresholdTime Interval ⌈

Parameter Name	IdsMEventThresholdTimeInterval	
Parent Container	IdsMEventThresholdFilter	
Upstream mapping defined	[5, FO_TPS_SecurityExtractTemplate]	
Description	Length of the threshold time interval (as float in seconds). Note: Shall be configured as a multiple of the IdsM main function period.	
Multiplicity	1	
Type	EcucFloatParamDef	
Range	[0 .. INF]	
Default value	–	
Pre-Compile only	true	

⌋

10.1.15 IdsMPdus

[ECUC_IdsM_00018] Definition of EcucParamConfContainerDef IdsMPdus ⌈

Container Name	IdsMPdus
Parent Container	IdsMConfiguration
Upstream mapping defined	–
Description	Configuration of the PDU references used to send the events data.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

No Included Parameters

Included Containers		
Container Name	**Multiplicity**	**Description**
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.
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.
IdsMTransmissionRetry	0..1	Configuration parameters for the transmission retry.

⌋

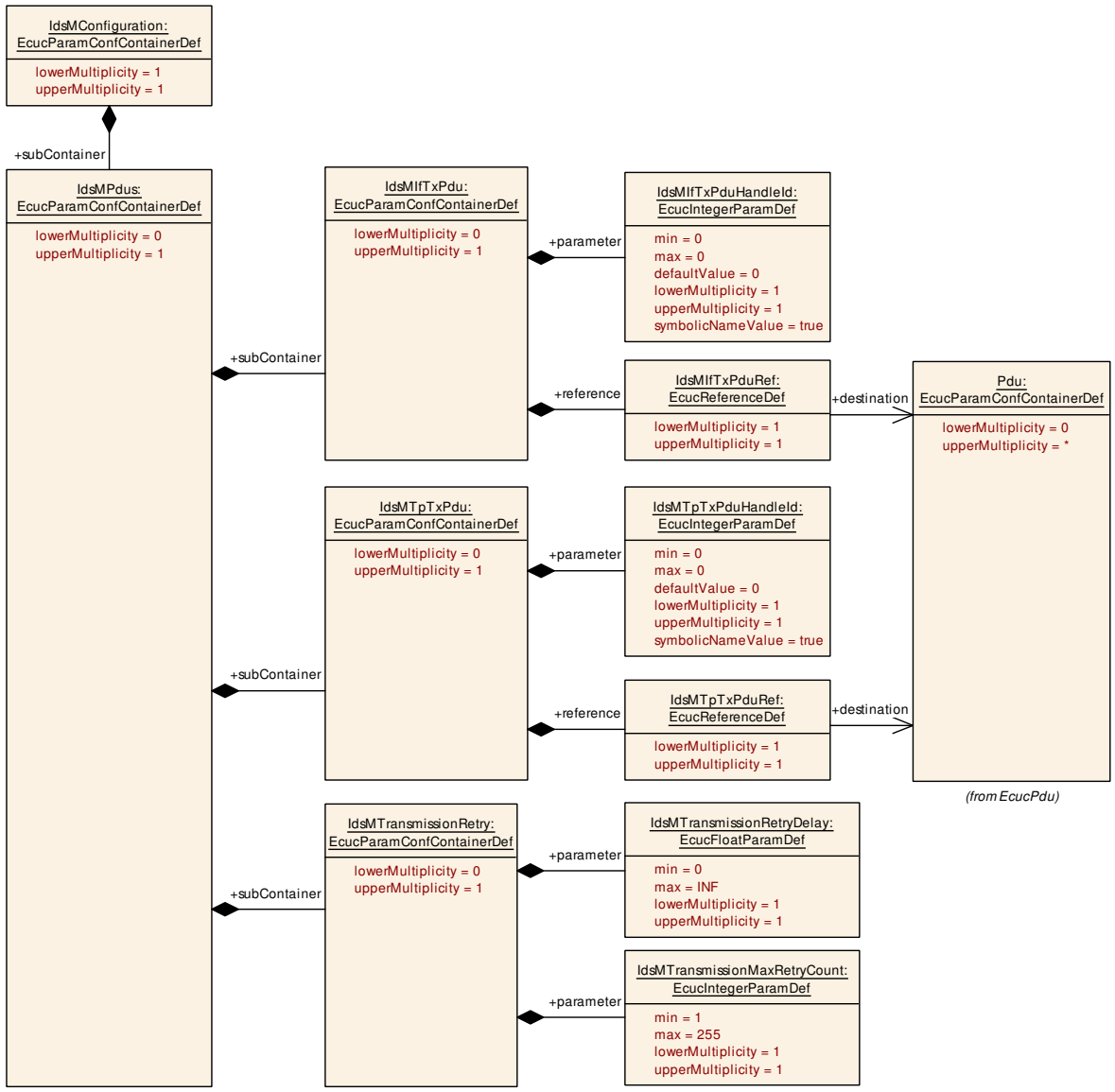


Figure 10.6: IdsM Pdus overview

10.1.16 IdsMIfTxPdu

[ECUC_IdsM_00040] Definition of EcucParamConfContainerDef IdsMIfTxPdu ⌈

Container Name	IdsMIfTxPdu
Parent Container	IdsMPdus
Upstream mapping defined	–
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.
Multiplicity	0..1

▽

△

Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
IdsMIfTxPduHandleId	1	[ECUC_IdsM_00041]
IdsMIfTxPduRef	1	[ECUC_IdsM_00042]

No Included Containers

⌋

[ECUC_IdsM_00041] Definition of EcucIntegerParamDef IdsMIfTxPduHandleId ⌈

Parameter Name	IdsMIfTxPduHandleId	
Parent Container	IdsMIfTxPdu	
Upstream mapping defined	–	
Description	IdsM does not use this parameter, content will be ignored. The existence of this parameter is needed by PduR.	
Multiplicity	1	
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)	
Range	0 .. 0	
Default value	0	
Pre-Compile only	true	

⌋

[ECUC_IdsM_00042] Definition of EcucReferenceDef IdsMIfTxPduRef ⌈

Parameter Name	IdsMIfTxPduRef	
Parent Container	IdsMIfTxPdu	
Upstream mapping defined	[7, CP_TPS_SystemTemplate]	
Description	Reference to the IF PDU used for transmission of the QSEvs.	
Multiplicity	1	
Type	Reference to Pdu	
Post-Build Multiplicity	-	Selectable
Post-Build Value	-	Selectable

⌋

10.1.17 IdsMEventTpTxPdu

[ECUC_IdsM_00043] Definition of EcucParamConfContainerDef IdsMTpTxPdu ⌈

Container Name	IdsMTpTxPdu
Parent Container	IdsMPdus
Upstream mapping defined	–
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.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
IdsMTpTxPduHandleId	1	[ECUC_IdsM_00044]
IdsMTpTxPduRef	1	[ECUC_IdsM_00045]

No Included Containers

⌋

[ECUC_IdsM_00044] Definition of EcucIntegerParamDef IdsMTpTxPduHandleId ⌈

Parameter Name	IdsMTpTxPduHandleId	
Parent Container	IdsMTpTxPdu	
Upstream mapping defined	–	
Description	IdsM does not use this parameter, content will be ignored. The existence of this parameter is needed by PduR.	
Multiplicity	1	
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)	
Range	0 .. 0	
Default value	0	
Pre-Compile only	true	

⌋

[ECUC_IdsM_00045] Definition of EcucReferenceDef IdsMTpTxPduRef ⌈

Parameter Name	IdsMTpTxPduRef	
Parent Container	IdsMTpTxPdu	
Upstream mapping defined	[7, CP_TPS_SystemTemplate]	
Description	Reference to the TP PDU used for transmission of the QSEvs.	
Multiplicity	1	
Type	Reference to Pdu	
Post-Build Multiplicity	-	Selectable
Post-Build Value	-	Selectable

⌋

10.1.18 IdsMTransmissionRetry

[ECUC_IdsM_00088] Definition of EcucParamConfContainerDef IdsMTransmissionRetry ⌈

Container Name	IdsMTransmissionRetry
Parent Container	IdsMPdus
Upstream mapping defined	–
Description	Configuration parameters for the transmission retry.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
IdsMTransmissionMaxRetryCount	1	[ECUC_IdsM_00090]
IdsMTransmissionRetryDelay	1	[ECUC_IdsM_00089]

No Included Containers

⌋

[ECUC_IdsM_00090] Definition of EcucIntegerParamDef IdsMTransmissionMaxRetryCount ⌈

Parameter Name	IdsMTransmissionMaxRetryCount
Parent Container	IdsMTransmissionRetry
Upstream mapping defined	–
Description	This parameter configures the number of retry attempt to be made for any failed transmission.
Multiplicity	1
Type	EcucIntegerParamDef
Range	1 .. 255
Default value	–
Pre-Compile only	true

⌋

[ECUC_IdsM_00089] Definition of EcucFloatParamDef IdsMTransmissionRetryDelay ⌈

Parameter Name	IdsMTransmissionRetryDelay
Parent Container	IdsMTransmissionRetry
Upstream mapping defined	–
Description	This parameter configures a delay time in seconds to be observed before a failed transmission is reattempted. The delay period shall be multiple of the IdsMMain FunctionPeriod.
Multiplicity	1
Type	EcucFloatParamDef

▽

△

Range	[0 .. INF]	
Default value	–	
Pre-Compile only	true	

⌋

10.1.19 IdsMBufferConfiguration

[ECUC_IdsM_00019] Definition of EcucParamConfContainerDef IdsMBufferConfiguration ⌈

Container Name	IdsMBufferConfiguration
Parent Container	IdsMConfiguration
Upstream mapping defined	–
Description	Configuration of the event buffers and context data buffers used by IdsM.
Multiplicity	1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
IdsMEventDisplacementStrategy	1	[ECUC_IdsM_00083]

Included Containers		
Container Name	**Multiplicity**	**Description**
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.
IdsMEventBuffers	1	Buffers used to store the SEvs.
IdsMQualifiedEventBuffers	1	Buffers used to store the QSEvs.

⌋

[ECUC_IdsM_00083] Definition of EcucEnumerationParamDef IdsMEventDisplacementStrategy ⌈

Parameter Name	IdsMEventDisplacementStrategy	
Parent Container	IdsMBufferConfiguration	
Upstream mapping defined	–	
Description	his parameter specifies the IdsM the displacement approach in the IdsMEventBuffer and IdsMQualifiedEventBuffers.	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	IDSM_BUFFER_DISPLACEMENT_DROP_LATEST	If the buffer is full and a new event is generated/ received, IdsM discards the newly generated/ received event.

▽

△

	IDSM_BUFFER_DISPLACEMENT_SEVERITY_BASED	If the buffer is full and a new event is generated/ received, IdsM discards the least severe event in the buffer and replaces it with the newly generated/received event. The implementation defines the behavior for tie breaking.
Pre-Compile only	true	

⌋

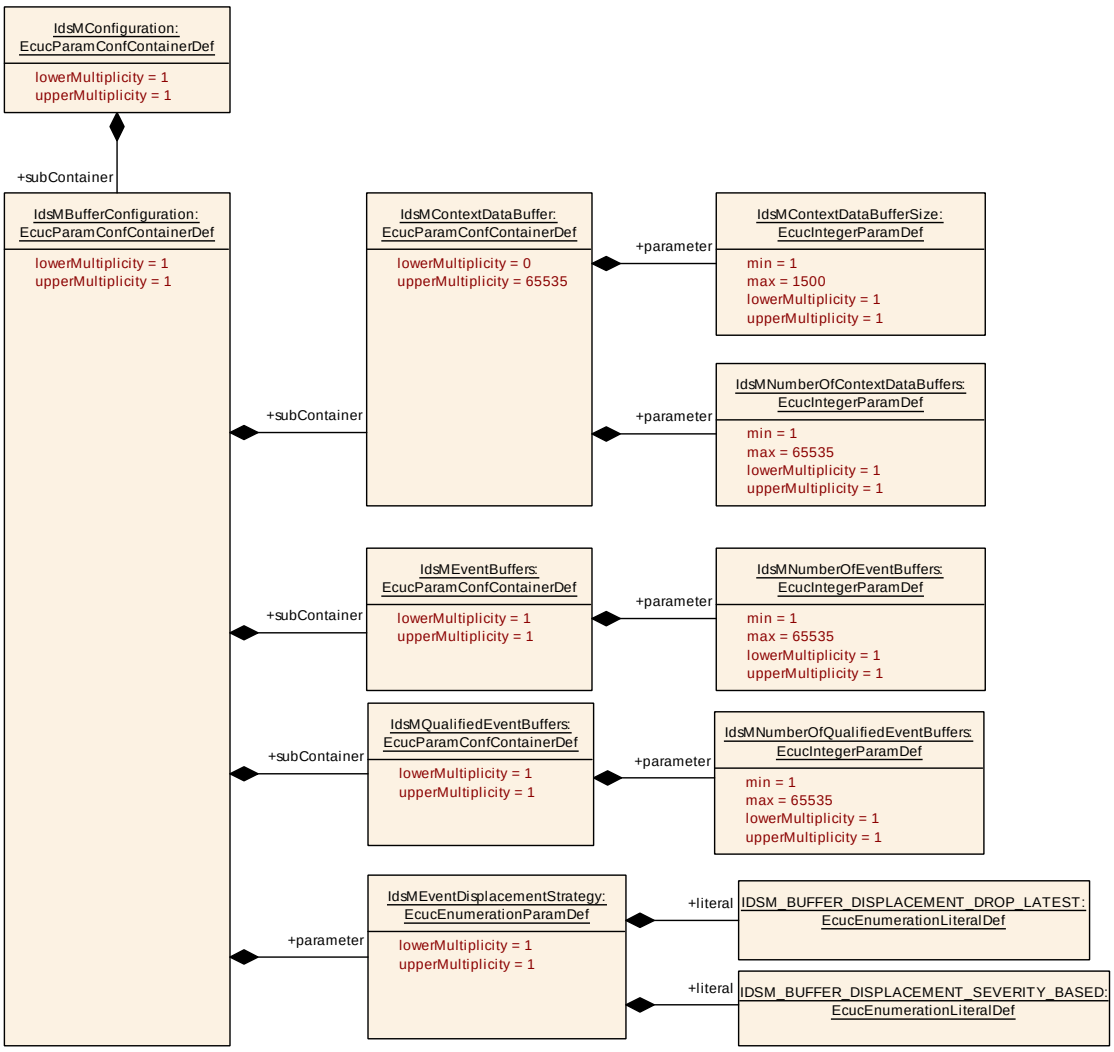


Figure 10.7: IdsM buffer configuration overview

10.1.20 IdsMContextDataBuffer

[ECUC_IdsM_00046] Definition of EcucParamConfContainerDef IdsMContext DataBuffer ⌈

Container Name	IdsMContextDataBuffer
Parent Container	IdsMBufferConfiguration
Upstream mapping defined	–
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.
Multiplicity	0..65535
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
IdsMContextDataBufferSize	1	[ECUC_IdsM_00047]
IdsMNumberOfContextDataBuffers	1	[ECUC_IdsM_00048]

No Included Containers

⌋

[ECUC_IdsM_00047] Definition of EcucIntegerParamDef IdsMContextDataBufferSize ⌈

Parameter Name	IdsMContextDataBufferSize	
Parent Container	IdsMContextDataBuffer	
Upstream mapping defined	–	
Description	Size of the context data buffer in bytes. It is recommended to configure buffers with an appropriate size depending on the configured SEvs.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	1 .. 1500	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_IdsM_00048] Definition of EcucIntegerParamDef IdsMNumberOfContextDataBuffers ⌈

Parameter Name	IdsMNumberOfContextDataBuffers	
Parent Container	IdsMContextDataBuffer	
Upstream mapping defined	–	
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.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	1 .. 65535	
Default value	–	
Pre-Compile only	true	

⌋

10.1.21 IdsMEventBuffers

[ECUC_IdsM_00049] Definition of EcucParamConfContainerDef IdsMEvent Buffers ⌈

Container Name	IdsMEventBuffers
Parent Container	IdsMBufferConfiguration
Upstream mapping defined	–
Description	Buffers used to store the SEvs.
Multiplicity	1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
IdsMNumberOfEventBuffers	1	[ECUC_IdsM_00050]

No Included Containers

⌋

[ECUC_IdsM_00050] Definition of EcucIntegerParamDef IdsMNumberOfEvent Buffers ⌈

Parameter Name	IdsMNumberOfEventBuffers
Parent Container	IdsMEventBuffers
Upstream mapping defined	–
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.
Multiplicity	1
Type	EcucIntegerParamDef
Range	1 .. 65535
Default value	–
Pre-Compile only	true

⌋

10.1.22 IdsMQualifiedEventBuffers

[ECUC_IdsM_00078] Definition of EcucParamConfContainerDef IdsMQualified EventBuffers ⌈

Container Name	IdsMQualifiedEventBuffers
Parent Container	IdsMBufferConfiguration
Upstream mapping defined	–
Description	Buffers used to store the QSEvs.
Multiplicity	1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
IdsMNumberOfQualifiedEventBuffers	1	[ECUC_IdsM_00080]

No Included Containers

⌋

[ECUC_IdsM_00080] Definition of EcucIntegerParamDef IdsMNumberOfQualifiedEventBuffers ⌈

Parameter Name	IdsMNumberOfQualifiedEventBuffers
Parent Container	IdsMQualifiedEventBuffers
Upstream mapping defined	–
Description	The number of qualified event buffers used to store the SEvs. Depending on the rate at which QSEvs are generated and the rate at which the sinks can consume them successfully, the number of QSEv buffers can be determined, e.g., assuming the sink SinkIdsR has a worst-case TxConfirmation latency of "k" milliseconds, if for every "n" millisecond period where "n" is considerably larger than "k", at the maximum "m" QSEv can be generated, IdsM would need at least "m" IdsMNumberOfQualifiedEventBuffers to prevent dropping QSEvs. The window "n" can be chosen as any interval considerably larger than "k" and where the peak QSEv generation rate can be estimated. Therefore IdsMNumberOfQualifiedEventBuffers >= n/k.
Multiplicity	1
Type	EcucIntegerParamDef
Range	1 .. 65535
Default value	–
Pre-Compile only	true

⌋

10.1.23 IdsMEvent

[ECUC_IdsM_00017] Definition of EcucParamConfContainerDef IdsMEvent ⌈

Container Name	IdsMEvent
Parent Container	IdsMConfiguration
Upstream mapping defined	[5, FO_TPS_SecurityExtractTemplate]
Description	Configuration of the IdsM Event unit which is reported by a sensor and its parameters.

▽

△

Multiplicity	1..65535
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
IdsMEventSeverity	0..1	[ECUC_IdsM_00082]
IdsMEventStatistics	1	[ECUC_IdsM_00096]
IdsMExternalEventId	1	[ECUC_IdsM_00032]
IdsMInternalEventId	1	[ECUC_IdsM_00033]
IdsMReportingModeFilter	1	[ECUC_IdsM_00036]
IdsMSensorInstanceId	1	[ECUC_IdsM_00031]
IdsMSinkDem	1	[ECUC_IdsM_00035]
IdsMSinkIdsR	1	[ECUC_IdsM_00034]
IdsMDemEventReference	0..1	[ECUC_IdsM_00086]
IdsMFilterChainRef	0..1	[ECUC_IdsM_00030]

Included Containers		
Container Name	**Multiplicity**	**Description**
IdsMContextDataModifierOptions	0..1	Adittional configuration parameters of a SEv to be able to modify incoming context data.
IdsMServiceInterfaceOptions	0..1	Adittional configuration parameters of a SEv when the sensor is a SW-C or application.

⌋

[ECUC_IdsM_00082] Definition of EcucIntegerParamDef IdsMEventSeverity ⌈

Parameter Name	IdsMEventSeverity
Parent Container	IdsMEvent
Upstream mapping defined	[5, FO_TPS_SecurityExtractTemplate]
Description	The severity level of a security event. The severity of the event is used for determining the event to be dropped when buffers are full. Severity 0 is interpreted as the lowest and 255 as the highest severity.
Multiplicity	0..1
Type	EcucIntegerParamDef
Range	0 .. 255
Default value	0
Pre-Compile only	true

⌋

[ECUC_IdsM_00096] Definition of EcucBooleanParamDef IdsMEventStatistics ⌈

Parameter Name	IdsMEventStatistics
Parent Container	IdsMEvent
Upstream mapping defined	–
Description	A boolean configuration to enable and disable collecting event specific statistics. .

▽

△

Multiplicity	1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

⌋

[ECUC_IdsM_00032] Definition of EcucIntegerParamDef IdsMExternalEventId ⌈

Parameter Name	IdsMExternalEventId
Parent Container	IdsMEvent
Upstream mapping defined	[5, FO_TPS_SecurityExtractTemplate]
Description	The external security event ID which is reported to the sink.There are two different value ranges depending on the referencing module: Standardized SEv ID is defined by the AUTOSAR specification.This ID is usually derived from the SecXT. Standard ID range: 0x0000 - 0x7FFF Generic User Event ID is defined by the user. Used when the SEv is not defined by the AUTOSAR specification, for example from a SW-C or a CDD. Generic ID range: 0x8000 - 0xFFFE. 0xFFFF is considered an invalid ID
Multiplicity	1
Type	EcucIntegerParamDef
Range	0 .. 65534
Default value	–
Pre-Compile only	true

⌋

[ECUC_IdsM_00033] Definition of EcucIntegerParamDef IdsMInternalEventId ⌈

Parameter Name	IdsMInternalEventId
Parent Container	IdsMEvent
Upstream mapping defined	–
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.
Multiplicity	1
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)
Range	0 .. 65535
Default value	65535
Pre-Compile only	true

⌋

[ECUC_IdsM_00036] Definition of EcucEnumerationParamDef IdsMReportingModeFilter ⌈

Parameter Name	IdsMReportingModeFilter
Parent Container	IdsMEvent
Upstream mapping defined	[5, FO_TPS_SecurityExtractTemplate]

▽

△

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)	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	BRIEF	SEv is sent without context data (BRIEF)
	BRIEF_BYPASSING_FILTERS	SEv is sent without context data, ignoring the rest of the filter chain (BRIEF_BYPASSING_FILTERS)
	DETAILED	SEv is sent with context data (DETAILED)
	DETAILED_BYPASSING_FILTERS	SEv is sent with context data ignoring the rest of the filter chain (DETAILED_BYPASSING_FILTERS)
	OFF	SEv is dropped (OFF)
Pre-Compile only	true	

⌋

[ECUC_IdsM_00031] Definition of EcucIntegerParamDef IdsMSensorInstanceId ⌈

Parameter Name	IdsMSensorInstanceId	
Parent Container	IdsMEvent	
Upstream mapping defined	[5, FO_TPS_SecurityExtractTemplate]	
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.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 63	
Default value	0	
Pre-Compile only	true	

⌋

[ECUC_IdsM_00035] Definition of EcucBooleanParamDef IdsMSinkDem ⌈

Parameter Name	IdsMSinkDem
Parent Container	IdsMEvent
Upstream mapping defined	[8, CP_TPS_DiagnosticExtractTemplate]
Description	The QSEv will be sent to the Dem Module into a Security Event Memory (Sem) to persist it on the local ECU.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

⌋

[ECUC_IdsM_00034] Definition of EcucBooleanParamDef IdsMSinkIdsR ⌈

Parameter Name	IdsMSinkIdsR
Parent Container	IdsMEvent
Upstream mapping defined	–
Description	The QSEv will be sent to the IDS Reporter.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

⌋

[ECUC_IdsM_00086] Definition of EcucReferenceDef IdsMDemEventReference ⌈

Parameter Name	IdsMDemEventReference
Parent Container	IdsMEvent
Upstream mapping defined	–
Description	Reference to DemEventParameter used for sinking QSEv into Dem.
Multiplicity	0..1
Type	Symbolic name reference to DemEventParameter
Pre-Compile only	true

⌋

[ECUC_IdsM_00030] Definition of EcucReferenceDef IdsMFilterChainRef ⌈

Parameter Name	IdsMFilterChainRef
Parent Container	IdsMEvent
Upstream mapping defined	[5, FO_TPS_SecurityExtractTemplate]
Description	Reference to a configured IdsM filter chain.
Multiplicity	0..1
Type	Reference to IdsMFilterChain
Pre-Compile only	true

⌋

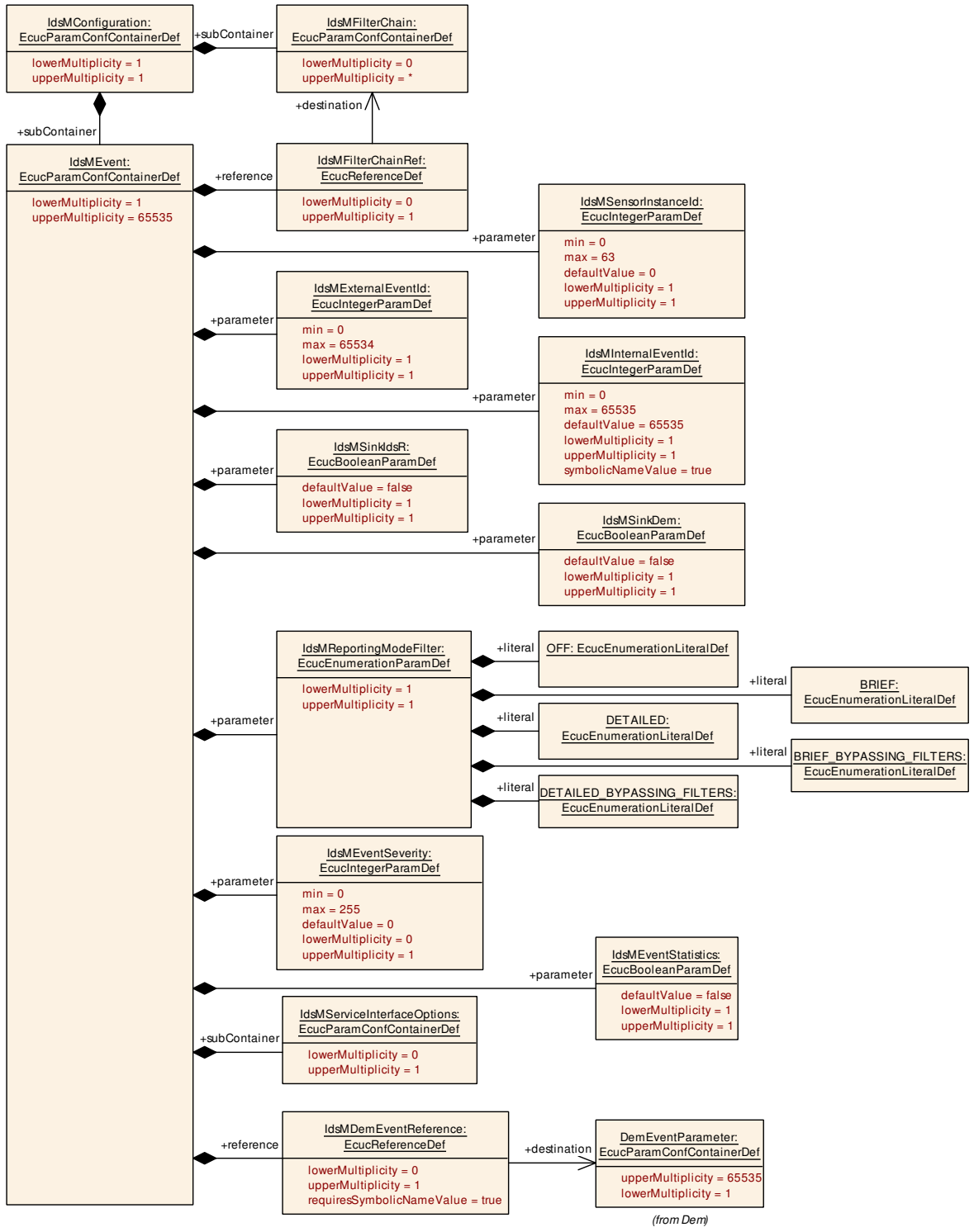


Figure 10.8: IdsM event overview

10.1.24 IdsMReportingModeFilter

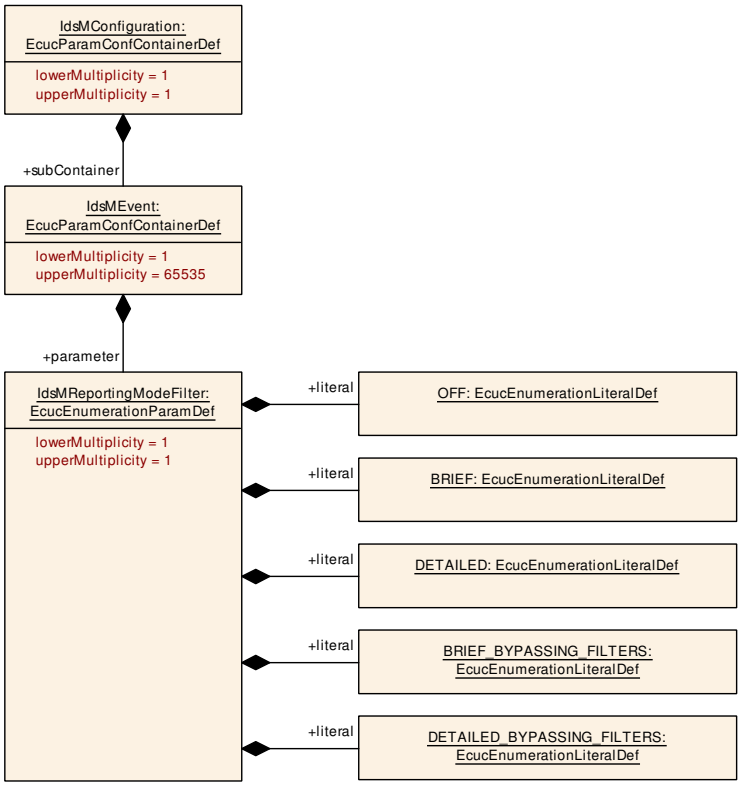


Figure 10.9: IdsM reporting mode filter overview

10.1.25 IdsMServiceInterfaceOptions

[ECUC_IdsM_00037] Definition of EcucParamConfContainerDef IdsMServiceInterfaceOptions ⌈

Container Name	IdsMServiceInterfaceOptions
Parent Container	IdsMEvent
Upstream mapping defined	–
Description	Adittional configuration parameters of a SEv when the sensor is a SW-C or application.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
IdsMEventEnableTimestampReporting	1	[ECUC_IdsM_00087]
IdsMEventMaxContextDataSize	1	[ECUC_IdsM_00038]

No Included Containers

⌋

[ECUC_IdsM_00087] Definition of EcucBooleanParamDef IdsMEventEnableTimestampReporting ⌈

Parameter Name	IdsMEventEnableTimestampReporting
Parent Container	IdsMServiceInterfaceOptions
Upstream mapping defined	–
Description	Enables/disables reporting of timestamps over C/S interfaces. • true: reporting is enabled. • false: reporting is disabled.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_IdsM_00038] Definition of EcucIntegerParamDef IdsMEventMaxContextDataSize ⌈

Parameter Name	IdsMEventMaxContextDataSize
Parent Container	IdsMServiceInterfaceOptions
Upstream mapping defined	–
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.
Multiplicity	1
Type	EcucIntegerParamDef
Range	0 .. 1500
Default value	0
Pre-Compile only	true

⌋

10.1.26 IdsMContextDataModifierOptions

[ECUC_IdsM_00061] Definition of EcucParamConfContainerDef IdsMContextDataModifierOptions ⌈

Container Name	IdsMContextDataModifierOptions
Parent Container	IdsMEvent
Upstream mapping defined	–
Description	Adittional configuration parameters of a SEv to be able to modify incoming context data.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
IdsMContextDataModifierCalloutRef	1	[ECUC_IdsM_00063]

No Included Containers

⌋

[ECUC_IdsM_00063] Definition of EcucReferenceDef IdsMContextDataModifierCalloutRef ⌈

Parameter Name	IdsMContextDataModifierCalloutRef
Parent Container	IdsMContextDataModifierOptions
Upstream mapping defined	–
Description	Reference to event specific context data modifier callout function.
Multiplicity	1
Type	Reference to IdsMContextDataModifierCallout
Pre-Compile only	true

⌋

10.1.27 IdsMBlockState

[ECUC_IdsM_00020] Definition of EcucParamConfContainerDef IdsMBlockState ⌈

Container Name	IdsMBlockState
Parent Container	IdsMConfiguration
Upstream mapping defined	[5, FO_TPS_SecurityExtractTemplate]
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().
Multiplicity	0..16
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
IdsMBlockStateID	1	[ECUC_IdsM_00052]

No Included Containers

⌋

[ECUC_IdsM_00052] Definition of EcucIntegerParamDef IdsMBlockStateID ⌈

Parameter Name	IdsMBlockStateID
Parent Container	IdsMBlockState
Upstream mapping defined	–
Description	This value specifies the identifier of this block state.
Multiplicity	1
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)
Range	0 .. 15
Default value	–
Pre-Compile only	true

⌋

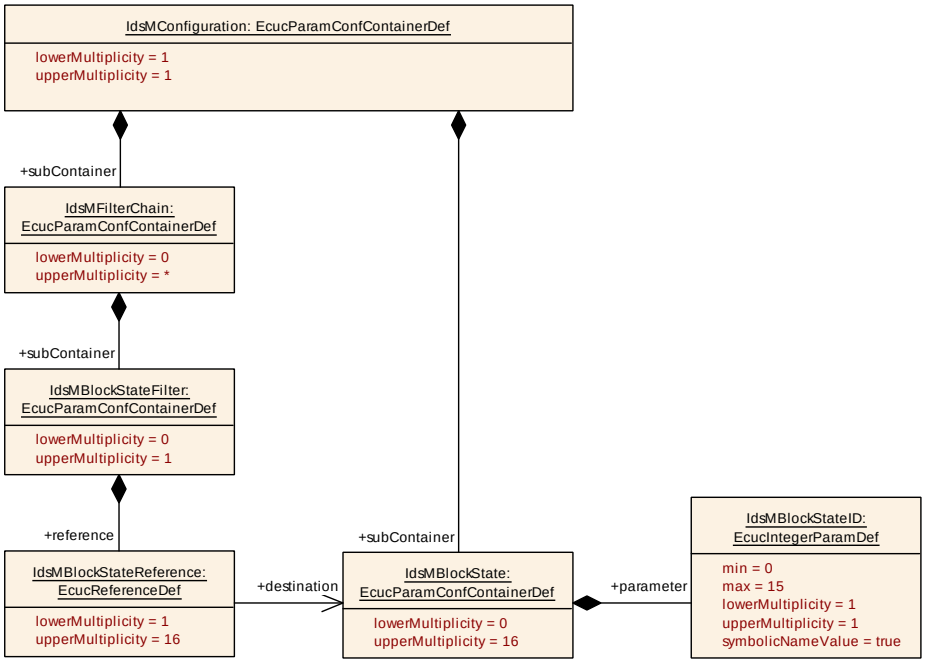


Figure 10.10: IdsM block state configuration overview

10.1.28 IdsMContextDataModification

[ECUC_IdsM_00064] Definition of EcucParamConfContainerDef IdsMContext DataModification ⌈

Container Name	IdsMContextDataModification
Parent Container	IdsMConfiguration
Upstream mapping defined	–
Description	Configuration of context data modifier callouts.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
IdsMGlobalContextDataHeaderFileIncludes	0..*	[ECUC_IdsM_00072]
IdsMGlobalContextDataModifierCalloutFct	0..1	[ECUC_IdsM_00070]
IdsMGlobalContextDataSizeModifier	0..1	[ECUC_IdsM_00071]

Included Containers		
Container Name	**Multiplicity**	**Description**
IdsMContextDataModifierCallout	0..65535	Configuration of event specific context data modifier callouts.

⌋

[ECUC_IdsM_00072] Definition of EcucStringParamDef IdsMGlobalContextData HeaderFileIncludes ⌈

Parameter Name	IdsMGlobalContextDataHeaderFileIncludes
Parent Container	IdsMContextDataModification
Upstream mapping defined	–
Description	List of optional header files which contains the prototypes of the context data modifier callouts.
Multiplicity	0..*
Type	EcucStringParamDef
Default value	–
Regular Expression	–
Pre-Compile only	true

⌋

[ECUC_IdsM_00070] Definition of EcucFunctionNameDef IdsMGlobalContext DataModifierCalloutFct ⌈

Parameter Name	IdsMGlobalContextDataModifierCalloutFct
Parent Container	IdsMContextDataModification
Upstream mapping defined	–
Description	Global context data modifier callout function. This callout can be overruled by event-specific context data modifier callouts.
Multiplicity	0..1
Type	EcucFunctionNameDef
Default value	–
Regular Expression	–
Pre-Compile only	true

⌋

[ECUC_IdsM_00071] Definition of EcucIntegerParamDef IdsMGlobalContextData SizeModifier ⌈

Parameter Name	IdsMGlobalContextDataSizeModifier
Parent Container	IdsMContextDataModification
Upstream mapping defined	–

▽

△

Description	Global context data size modifier used in combination with global context data size modifier callout function.	
Multiplicity	0..1	
Type	EcucIntegerParamDef	
Range	-1499 .. 1499	
Default value	0	
Pre-Compile only	true	

┘

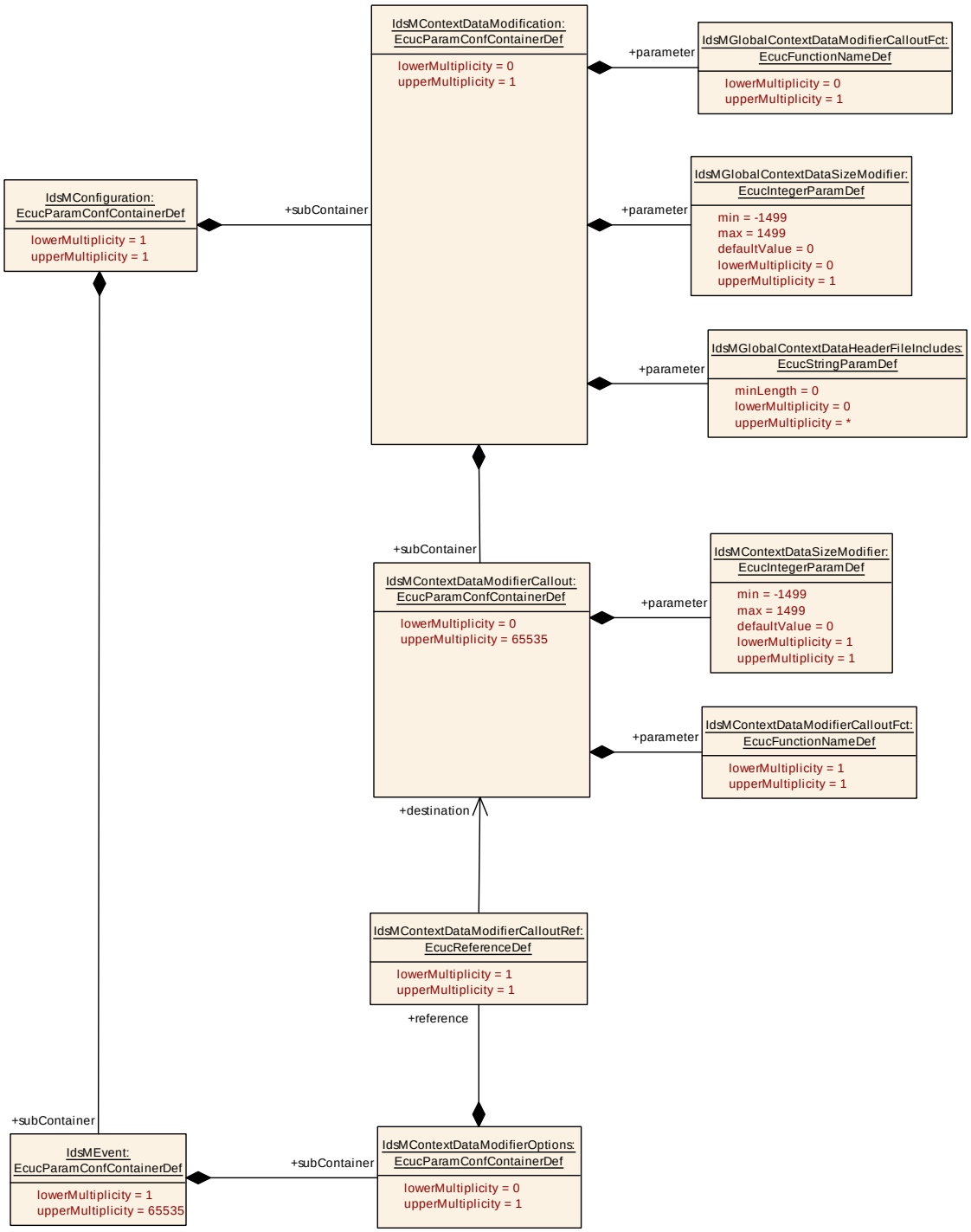


Figure 10.11: IdsM context data modification configuration overview

10.2 Configuration Constraints

This section lists configuration constraints for the `IdsM` Module.

[SWS_IdsM_CONSTR_00002] Minimum one filter must be configured ⌈Instances of the container IdsMFilterChain always require to have at least one filter configured (IdsMBlockStateFilter, IdsMForwardEveryNthFilter, IdsMEventAggregationFilter, IdsMEventThresholdFilter).

⌋

[SWS_IdsM_CONSTR_00003] References between IdsMEvent and DemEvent required ⌈Instances of the container IdsMEvent shall require IdsMDemEventReference when the IdsMSinkDem is TRUE.⌋

[SWS_IdsM_CONSTR_00004] Maximum Context Data Size equal or less than largest Context Data Buffer ⌈IdsMEventMaxContextDataSize of each IdsMEvent shall be lesser than or equal to the largest IdsMContextDataBufferSize.⌋

[SWS_IdsM_CONSTR_00005] Reference Csm job primitive to Csm job ⌈The CsmPrimitive referenced by the CsmJob used by IdsMCsmJobReference shall be either Csm_MacGenerate or Csm_SignatureGenerate.⌋

[SWS_IdsM_CONSTR_00006] Authenticator length dependent to Signature Generate Result ⌈The value of IdsMAuthenticatorLength shall be larger than or equal to signatureLength when IdsMCsmJobReference resolves to a CsmJob whose CsmPrimitive includes a Csm_SignatureGenerate container.⌋

[SWS_IdsM_CONSTR_00007] Consideration of context data buffer size for IdsM statistics ⌈If the event SEV_IDSM_STATISTICS is configured, IdsM shall ensure that a sufficiently large context data buffer is available to hold the variable context data size generated by IdsM. The variable context data is defined on the chapter on 7.11.⌋

10.3 Published Information

For details refer to [2] Chapter 10.3 *"Published Information"*.

A Change history of AUTOSAR traceable items

Please note that the lists in this chapter also include traceable items that have been removed from the specification in a later version. These items do not appear as hyperlinks in the document.

A.1 Traceable item history of this document according to AUTOSAR Release R26-11

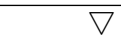
A.1.1 Added Specification Items in R26-11

Number	Heading
[ECUC_IdsM_00093]	Definition of EcucParamConfContainerDef IdsMStatistics
[ECUC_IdsM_00094]	Definition of EcucBooleanParamDef IdsMInstanceStatistics
[ECUC_IdsM_00095]	Definition of EcucBooleanParamDef IdsMPersistStatistics
[ECUC_IdsM_00096]	Definition of EcucBooleanParamDef IdsMEventStatistics
[ECUC_IdsM_00097]	Definition of EcucReferenceDef SEV_IDSM_STATISTICS
[SWS_IdsM_03004]	Global Count Security Events
[SWS_IdsM_03005]	Global Count Qualified Security Events
[SWS_IdsM_03006]	Global Count Transmitted Security Events
[SWS_IdsM_03007]	Global Count Failed Transmitted Security Events
[SWS_IdsM_03008]	Global Count Persisted Security Events
[SWS_IdsM_03009]	Event Count Received Security Events
[SWS_IdsM_03010]	Event Count Qualified Security Events
[SWS_IdsM_03011]	Peak Fill Status Event Buffer
[SWS_IdsM_03012]	Peak Fill Status Size Context Data Buffer
[SWS_IdsM_03013]	Peak Fill Status Size Qualified Event Buffer
[SWS_IdsM_03014]	Persist Statistics
[SWS_IdsM_03015]	Add Statistics to Context Data
[SWS_IdsM_03016]	Security event context data definition: SEV_IDSM_STATISTICS
[SWS_IdsM_03017]	Statistics Control Field
[SWS_IdsM_03018]	Instance Statistics Buffers
[SWS_IdsM_03019]	Event Specific Statistics
[SWS_IdsM_03020]	Drop

Table A.1: Added Specification Items in R26-11

A.1.2 Changed Specification Items in R26-11

Number	Heading
[ECUC_IdsM_00001]	Definition of EcucModuleDef IdsM
[ECUC_IdsM_00002]	Definition of EcucParamConfContainerDef IdsMGeneral
[ECUC_IdsM_00003]	Definition of EcucParamConfContainerDef IdsMConfiguration
[ECUC_IdsM_00004]	Definition of EcucFloatParamDef IdsMMainFunctionPeriod
[ECUC_IdsM_00005]	Definition of EcucBooleanParamDef IdsMDevErrorDetect
[ECUC_IdsM_00007]	Definition of EcucIntegerParamDef IdsMInstanceId
[ECUC_IdsM_00008]	Definition of EcucParamConfContainerDef IdsMGlobalRateLimitationFilters
[ECUC_IdsM_00009]	Definition of EcucBooleanParamDef IdsMAuthenticatorSupport
[ECUC_IdsM_00010]	Definition of EcucBooleanParamDef IdsMDiagnosticSupport
[ECUC_IdsM_00011]	Definition of EcucIntegerParamDef IdsMAuthenticatorLength
[ECUC_IdsM_00012]	Definition of EcucEnumerationParamDef IdsMTimestampOption
[ECUC_IdsM_00013]	Definition of EcucReferenceDef IdsMNvmBlockDescriptor
[ECUC_IdsM_00014]	Definition of EcucReferenceDef IdsMStbmTimeBaseReference
[ECUC_IdsM_00015]	Definition of EcucReferenceDef IdsMCsmJobReference
[ECUC_IdsM_00016]	Definition of EcucParamConfContainerDef IdsMFilterChain
[ECUC_IdsM_00017]	Definition of EcucParamConfContainerDef IdsMEvent
[ECUC_IdsM_00018]	Definition of EcucParamConfContainerDef IdsMPdus
[ECUC_IdsM_00019]	Definition of EcucParamConfContainerDef IdsMBufferConfiguration
[ECUC_IdsM_00020]	Definition of EcucParamConfContainerDef IdsMBlockState
[ECUC_IdsM_00021]	Definition of EcucParamConfContainerDef IdsMBlockStateFilter
[ECUC_IdsM_00022]	Definition of EcucParamConfContainerDef IdsMForwardEveryNthFilter
[ECUC_IdsM_00023]	Definition of EcucIntegerParamDef IdsMNthParameter
[ECUC_IdsM_00024]	Definition of EcucParamConfContainerDef IdsMEventAggregationFilter
[ECUC_IdsM_00025]	Definition of EcucFloatParamDef IdsMEventAggregationTimeInterval
[ECUC_IdsM_00026]	Definition of EcucEnumerationParamDef IdsMContextDataSourceSelector
[ECUC_IdsM_00027]	Definition of EcucParamConfContainerDef IdsMEventThresholdFilter
[ECUC_IdsM_00028]	Definition of EcucFloatParamDef IdsMEventThresholdTimeInterval
[ECUC_IdsM_00029]	Definition of EcucIntegerParamDef IdsMEventThresholdNumber
[ECUC_IdsM_00030]	Definition of EcucReferenceDef IdsMFilterChainRef
[ECUC_IdsM_00031]	Definition of EcucIntegerParamDef IdsMSensorInstanceId
[ECUC_IdsM_00032]	Definition of EcucIntegerParamDef IdsMExternalEventId
[ECUC_IdsM_00033]	Definition of EcucIntegerParamDef IdsMInternalEventId
[ECUC_IdsM_00034]	Definition of EcucBooleanParamDef IdsMSinkIdsR
[ECUC_IdsM_00035]	Definition of EcucBooleanParamDef IdsMSinkDem
[ECUC_IdsM_00036]	Definition of EcucEnumerationParamDef IdsMReportingModeFilter
[ECUC_IdsM_00037]	Definition of EcucParamConfContainerDef IdsMServiceInterfaceOptions



△

Number	Heading
[ECUC_IdsM_00038]	Definition of EcucIntegerParamDef IdsMEventMaxContextDataSize
[ECUC_IdsM_00040]	Definition of EcucParamConfContainerDef IdsMIfTxPdu
[ECUC_IdsM_00041]	Definition of EcucIntegerParamDef IdsMIfTxPduHandleId
[ECUC_IdsM_00042]	Definition of EcucReferenceDef IdsMIfTxPduRef
[ECUC_IdsM_00043]	Definition of EcucParamConfContainerDef IdsMTpTxPdu
[ECUC_IdsM_00044]	Definition of EcucIntegerParamDef IdsMTpTxPduHandleId
[ECUC_IdsM_00045]	Definition of EcucReferenceDef IdsMTpTxPduRef
[ECUC_IdsM_00046]	Definition of EcucParamConfContainerDef IdsMContextDataBuffer
[ECUC_IdsM_00047]	Definition of EcucIntegerParamDef IdsMContextDataBufferSize
[ECUC_IdsM_00048]	Definition of EcucIntegerParamDef IdsMNumberOfContextDataBuffers
[ECUC_IdsM_00049]	Definition of EcucParamConfContainerDef IdsMEventBuffers
[ECUC_IdsM_00050]	Definition of EcucIntegerParamDef IdsMNumberOfEventBuffers
[ECUC_IdsM_00051]	Definition of EcucReferenceDef IdsMBlockStateReference
[ECUC_IdsM_00052]	Definition of EcucIntegerParamDef IdsMBlockStateID
[ECUC_IdsM_00053]	Definition of EcucParamConfContainerDef IdsMFilterEventRateLimitation
[ECUC_IdsM_00054]	Definition of EcucFloatParamDef IdsMRateLimitationTimeInterval
[ECUC_IdsM_00055]	Definition of EcucIntegerParamDef IdsMRateLimitationMaximumEvents
[ECUC_IdsM_00056]	Definition of EcucParamConfContainerDef IdsMFilterTrafficLimitation
[ECUC_IdsM_00057]	Definition of EcucFloatParamDef IdsMTrafficLimitationTimeInterval
[ECUC_IdsM_00058]	Definition of EcucIntegerParamDef IdsMTrafficLimitationMaximumBytes
[ECUC_IdsM_00059]	Definition of EcucParamConfContainerDef IdsMAuthenticator
[ECUC_IdsM_00060]	Definition of EcucParamConfContainerDef IdsMTimestamp
[ECUC_IdsM_00061]	Definition of EcucParamConfContainerDef IdsMContextDataModifierOptions
[ECUC_IdsM_00063]	Definition of EcucReferenceDef IdsMContextDataModifierCalloutRef
[ECUC_IdsM_00064]	Definition of EcucParamConfContainerDef IdsMContextDataModification
[ECUC_IdsM_00065]	Definition of EcucParamConfContainerDef IdsMContextDataModifierCallout
[ECUC_IdsM_00066]	Definition of EcucIntegerParamDef IdsMContextDataSizeModifier
[ECUC_IdsM_00067]	Definition of EcucFunctionNameDef IdsMContextDataModifierCalloutFct
[ECUC_IdsM_00070]	Definition of EcucFunctionNameDef IdsMGlobalContextDataModifierCallout Fct
[ECUC_IdsM_00071]	Definition of EcucIntegerParamDef IdsMGlobalContextDataSizeModifier
[ECUC_IdsM_00072]	Definition of EcucStringParamDef IdsMGlobalContextDataHeaderFile Includes
[ECUC_IdsM_00073]	Definition of EcucParamConfContainerDef IdsMSecurityEventRefs
[ECUC_IdsM_00074]	Definition of EcucReferenceDef SEV_IDSM_NO_EVENT_BUFFER_ AVAILABLE
[ECUC_IdsM_00075]	Definition of EcucReferenceDef SEV_IDSM_NO_CONTEXT_DATA_ BUFFER_AVAILABLE
[ECUC_IdsM_00077]	Definition of EcucReferenceDef SEV_IDSM_COMMUNICATION_ERROR
[ECUC_IdsM_00078]	Definition of EcucParamConfContainerDef IdsMQualifiedEventBuffers

▽

△

Number	Heading
[ECUC_IdsM_00080]	Definition of EcucIntegerParamDef IdsMNumberOfQualifiedEventBuffers
[ECUC_IdsM_00081]	Definition of EcucReferenceDef SEV_IDSM_NO_QUALIFIED_EVENT_BUFFER_AVAILABLE
[ECUC_IdsM_00082]	Definition of EcucIntegerParamDef IdsMEventSeverity
[ECUC_IdsM_00083]	Definition of EcucEnumerationParamDef IdsMEventDisplacementStrategy
[ECUC_IdsM_00084]	Definition of EcucBooleanParamDef IdsMTimestampSupport
[ECUC_IdsM_00085]	Definition of EcucBooleanParamDef IdsMEnableSecurityEventReporting
[ECUC_IdsM_00086]	Definition of EcucReferenceDef IdsMDemEventReference
[ECUC_IdsM_00087]	Definition of EcucBooleanParamDef IdsMEventEnableTimestampReporting
[ECUC_IdsM_00088]	Definition of EcucParamConfContainerDef IdsMTransmissionRetry
[ECUC_IdsM_00089]	Definition of EcucFloatParamDef IdsMTransmissionRetryDelay
[ECUC_IdsM_00090]	Definition of EcucIntegerParamDef IdsMTransmissionMaxRetryCount
[ECUC_IdsM_00092]	Definition of EcucBooleanParamDef IdsMReportingModeApi
[SWS_IdsM_01304]	NULL-Pointer valid for parameter for ContextData-Pointer
[SWS_IdsM_01305]	NULL-Pointer valid for parameter for Timestamp-Pointer
[SWS_IdsM_91044]	Definition of ClientServerInterface IdsMService_{EventName}

Table A.2: Changed Specification Items in R26-11

A.1.3 Deleted Specification Items in R26-11

Number	Heading
[ECUC_IdsM_00006]	Definition of EcucBooleanParamDef IdsMVersionInfoApi
[ECUC_IdsM_00076]	Definition of EcucReferenceDef SEV_IDSM_TRAFFIC_LIMITATION_EXCEEDED
[ECUC_IdsM_00091]	Definition of EcucEnumerationParamDef IdsMAuthenticatorType
[SWS_IdsM_01094]	
[SWS_IdsM_91004]	Definition of API function IdsM_GetVersionInfo
[SWS_IdsM_91040]	Definition of ImplementationDataType IdsM_ContextDataPointerType
[SWS_IdsM_91041]	Definition of ImplementationDataType IdsM_TimestampPointerType
[SWS_IdsM_91045]	Definition of ClientServerInterface IdsMService_{EventName}
[SWS_IdsM_91046]	Definition of ClientServerInterface IdsMService_{EventName}
[SWS_IdsM_91047]	Definition of ClientServerInterface IdsMService_{EventName}

Table A.3: Deleted Specification Items in R26-11

A.1.4 Added Constraints in R26-11

Number	Heading
[SWS_IdsM_CONSTR_00005]	Reference Csm job primitive to Csm job
[SWS_IdsM_CONSTR_00006]	Authenticator length dependent to Signature Generate Result
[SWS_IdsM_CONSTR_00007]	Consideration of context data buffer size for IdsM statistics

Table A.4: Added Constraints in R26-11

A.1.5 Changed Constraints in R26-11

none

A.1.6 Deleted Constraints in R26-11

none

A.2 Traceable item history of this document according to AUTOSAR Release R25-11

A.2.1 Added Specification Items in R25-11

Number	Heading
[ECUC_IdsM_00088]	Definition of EcucParamConfContainerDef IdsMTransmissionRetry
[ECUC_IdsM_00089]	Definition of EcucFloatParamDef IdsMTransmissionRetryDelay
[ECUC_IdsM_00090]	Definition of EcucIntegerParamDef IdsMTransmissionMaxRetryCount
[ECUC_IdsM_00091]	Definition of EcucEnumerationParamDef IdsMAuthenticatorType
[ECUC_IdsM_00092]	Definition of EcucBooleanParamDef IdsMReportingModeApi
[SWS_IdsM_00914]	SEv count overflow
[SWS_IdsM_01503]	Retry Handling
[SWS_IdsM_02030]	Reporting Mode
[SWS_IdsM_02031]	Modify Reporting Mode
[SWS_IdsM_02032]	Reading Reporting Mode
[SWS_IdsM_02033]	Invalid Parameters

▽

△

Number	Heading
[SWS_IdsM_02034]	Busy Failure
[SWS_IdsM_02035]	Internal Error
[SWS_IdsM_02036]	Polling of Set Reporting Mode Operation
[SWS_IdsM_91048]	Definition of ImplementationDataType IdsM_ReportingModeStatusType
[SWS_IdsM_91049]	Definition of ImplementationDataType IdsM_SensorInstanceIdType
[SWS_IdsM_91050]	Definition of Std_ReturnType-extension for module IdsM
[SWS_IdsM_91051]	Definition of API function IdsM_GetReportingMode
[SWS_IdsM_91052]	Definition of API function IdsM_GetStatusReportingMode
[SWS_IdsM_91053]	Definition of API function IdsM_SetReportingMode
[SWS_IdsM_91054]	Definition of ClientServerInterface IdsMReportingModeService
[SWS_IdsM_91055]	Definition of Port IdsM_ReportingModeStatus provided by module IdsM

Table A.5: Added Specification Items in R25-11

A.2.2 Changed Specification Items in R25-11

Number	Heading
[ECUC_IdsM_00001]	Definition of EcucModuleDef IdsM
[ECUC_IdsM_00002]	Definition of EcucParamConfContainerDef IdsMGeneral
[ECUC_IdsM_00003]	Definition of EcucParamConfContainerDef IdsMConfiguration
[ECUC_IdsM_00004]	Definition of EcucFloatParamDef IdsMMainFunctionPeriod
[ECUC_IdsM_00005]	Definition of EcucBooleanParamDef IdsMDevErrorDetect
[ECUC_IdsM_00006]	Definition of EcucBooleanParamDef IdsMVersionInfoApi
[ECUC_IdsM_00007]	Definition of EcucIntegerParamDef IdsMInstanceId
[ECUC_IdsM_00008]	Definition of EcucParamConfContainerDef IdsMGlobalRateLimitationFilters
[ECUC_IdsM_00009]	Definition of EcucBooleanParamDef IdsMAuthenticatorSupport
[ECUC_IdsM_00010]	Definition of EcucBooleanParamDef IdsMDiagnosticSupport
[ECUC_IdsM_00011]	Definition of EcucIntegerParamDef IdsMAuthenticatorLength
[ECUC_IdsM_00012]	Definition of EcucEnumerationParamDef IdsMTimestampOption
[ECUC_IdsM_00013]	Definition of EcucReferenceDef IdsMNvmBlockDescriptor
[ECUC_IdsM_00014]	Definition of EcucReferenceDef IdsMStbmTimeBaseReference
[ECUC_IdsM_00015]	Definition of EcucReferenceDef IdsMCsmJobReference
[ECUC_IdsM_00016]	Definition of EcucParamConfContainerDef IdsMFilterChain
[ECUC_IdsM_00017]	Definition of EcucParamConfContainerDef IdsMEvent
[ECUC_IdsM_00018]	Definition of EcucParamConfContainerDef IdsMPdus
[ECUC_IdsM_00019]	Definition of EcucParamConfContainerDef IdsMBufferConfiguration
[ECUC_IdsM_00020]	Definition of EcucParamConfContainerDef IdsMBlockState

▽

△

Number	Heading
[ECUC_IdsM_00021]	Definition of EcucParamConfContainerDef IdsMBlockStateFilter
[ECUC_IdsM_00022]	Definition of EcucParamConfContainerDef IdsMForwardEveryNthFilter
[ECUC_IdsM_00023]	Definition of EcucIntegerParamDef IdsMNthParameter
[ECUC_IdsM_00024]	Definition of EcucParamConfContainerDef IdsMEventAggregationFilter
[ECUC_IdsM_00025]	Definition of EcucFloatParamDef IdsMEventAggregationTimeInterval
[ECUC_IdsM_00026]	Definition of EcucEnumerationParamDef IdsMContextDataSourceSelector
[ECUC_IdsM_00027]	Definition of EcucParamConfContainerDef IdsMEventThresholdFilter
[ECUC_IdsM_00028]	Definition of EcucFloatParamDef IdsMEventThresholdTimeInterval
[ECUC_IdsM_00029]	Definition of EcucIntegerParamDef IdsMEventThresholdNumber
[ECUC_IdsM_00030]	Definition of EcucReferenceDef IdsMFilterChainRef
[ECUC_IdsM_00031]	Definition of EcucIntegerParamDef IdsMSensorInstanceId
[ECUC_IdsM_00032]	Definition of EcucIntegerParamDef IdsMExternalEventId
[ECUC_IdsM_00033]	Definition of EcucIntegerParamDef IdsMInternalEventId
[ECUC_IdsM_00034]	Definition of EcucBooleanParamDef IdsMSinkIdsR
[ECUC_IdsM_00035]	Definition of EcucBooleanParamDef IdsMSinkDem
[ECUC_IdsM_00036]	Definition of EcucEnumerationParamDef IdsMReportingModeFilter
[ECUC_IdsM_00037]	Definition of EcucParamConfContainerDef IdsMServiceInterfaceOptions
[ECUC_IdsM_00038]	Definition of EcucIntegerParamDef IdsMEventMaxContextDataSize
[ECUC_IdsM_00040]	Definition of EcucParamConfContainerDef IdsMIfTxPdu
[ECUC_IdsM_00041]	Definition of EcucIntegerParamDef IdsMIfTxPduHandleId
[ECUC_IdsM_00042]	Definition of EcucReferenceDef IdsMIfTxPduRef
[ECUC_IdsM_00043]	Definition of EcucParamConfContainerDef IdsMTpTxPdu
[ECUC_IdsM_00044]	Definition of EcucIntegerParamDef IdsMTpTxPduHandleId
[ECUC_IdsM_00045]	Definition of EcucReferenceDef IdsMTpTxPduRef
[ECUC_IdsM_00046]	Definition of EcucParamConfContainerDef IdsMContextDataBuffer
[ECUC_IdsM_00047]	Definition of EcucIntegerParamDef IdsMContextDataBufferSize
[ECUC_IdsM_00048]	Definition of EcucIntegerParamDef IdsMNumberOfContextDataBuffers
[ECUC_IdsM_00049]	Definition of EcucParamConfContainerDef IdsMEventBuffers
[ECUC_IdsM_00050]	Definition of EcucIntegerParamDef IdsMNumberOfEventBuffers
[ECUC_IdsM_00051]	Definition of EcucReferenceDef IdsMBlockStateReference
[ECUC_IdsM_00052]	Definition of EcucIntegerParamDef IdsMBlockStateID
[ECUC_IdsM_00053]	Definition of EcucParamConfContainerDef IdsMFilterEventRateLimitation
[ECUC_IdsM_00054]	Definition of EcucFloatParamDef IdsMRateLimitationTimeInterval
[ECUC_IdsM_00055]	Definition of EcucIntegerParamDef IdsMRateLimitationMaximumEvents
[ECUC_IdsM_00056]	Definition of EcucParamConfContainerDef IdsMFilterTrafficLimitation
[ECUC_IdsM_00057]	Definition of EcucFloatParamDef IdsMTrafficLimitationTimeInterval
[ECUC_IdsM_00058]	Definition of EcucIntegerParamDef IdsMTrafficLimitationMaximumBytes
[ECUC_IdsM_00059]	Definition of EcucParamConfContainerDef IdsMAuthenticator

▽

△

Number	Heading
[ECUC_IdsM_00060]	Definition of EcucParamConfContainerDef IdsMTimestamp
[ECUC_IdsM_00061]	Definition of EcucParamConfContainerDef IdsMContextDataModifierOptions
[ECUC_IdsM_00063]	Definition of EcucReferenceDef IdsMContextDataModifierCalloutRef
[ECUC_IdsM_00064]	Definition of EcucParamConfContainerDef IdsMContextDataModification
[ECUC_IdsM_00065]	Definition of EcucParamConfContainerDef IdsMContextDataModifierCallout
[ECUC_IdsM_00066]	Definition of EcucIntegerParamDef IdsMContextDataSizeModifier
[ECUC_IdsM_00067]	Definition of EcucFunctionNameDef IdsMContextDataModifierCalloutFct
[ECUC_IdsM_00070]	Definition of EcucFunctionNameDef IdsMGlobalContextDataModifierCallout Fct
[ECUC_IdsM_00071]	Definition of EcucIntegerParamDef IdsMGlobalContextDataSizeModifier
[ECUC_IdsM_00072]	Definition of EcucStringParamDef IdsMGlobalContextDataHeaderFile Includes
[ECUC_IdsM_00073]	Definition of EcucParamConfContainerDef IdsMSecurityEventRefs
[ECUC_IdsM_00074]	Definition of EcucReferenceDef SEV_IDSM_NO_EVENT_BUFFER_ AVAILABLE
[ECUC_IdsM_00075]	Definition of EcucReferenceDef SEV_IDSM_NO_CONTEXT_DATA_ BUFFER_AVAILABLE
[ECUC_IdsM_00076]	Definition of EcucReferenceDef SEV_IDSM_TRAFFIC_LIMITATION_ EXCEEDED
[ECUC_IdsM_00077]	Definition of EcucReferenceDef SEV_IDSM_COMMUNICATION_ERROR
[ECUC_IdsM_00078]	Definition of EcucParamConfContainerDef IdsMQualifiedEventBuffers
[ECUC_IdsM_00080]	Definition of EcucIntegerParamDef IdsMNumberOfQualifiedEventBuffers
[ECUC_IdsM_00081]	Definition of EcucReferenceDef SEV_IDSM_NO_QUALIFIED_EVENT_ BUFFER_AVAILABLE
[ECUC_IdsM_00082]	Definition of EcucIntegerParamDef IdsMEventSeverity
[ECUC_IdsM_00083]	Definition of EcucEnumerationParamDef IdsMEventDisplacementStrategy
[ECUC_IdsM_00084]	Definition of EcucBooleanParamDef IdsMTimestampSupport
[ECUC_IdsM_00085]	Definition of EcucBooleanParamDef IdsMEnableSecurityEventReporting
[ECUC_IdsM_00086]	Definition of EcucReferenceDef IdsMDemEventReference
[ECUC_IdsM_00087]	Definition of EcucBooleanParamDef IdsMEventEnableTimestampReporting
[SWS_IdsM_00203]	Error UnInit
[SWS_IdsM_01034]	
[SWS_IdsM_01108]	Timestamp provided via event reporting interface
[SWS_IdsM_01111]	
[SWS_IdsM_01200]	
[SWS_IdsM_01204]	Authenticating QSEv
[SWS_IdsM_01205]	AuthenticatorLength
[SWS_IdsM_01206]	GenerateAuthenticator
[SWS_IdsM_01207]	SynchronousAuthenticatorGeneration
[SWS_IdsM_01208]	AsynchronousAuthenticatorGeneration

▽

△

Number	Heading
[SWS_IdsM_01400]	IF-PDU
[SWS_IdsM_01901]	Modify Reporting Mode via Diagnostics
[SWS_IdsM_91013]	Definition of ImplementationDataType IdsM_Filters_ReportingModeType
[SWS_IdsM_91017]	Definition of datatype IdsM_Filters_BlockStateType
[SWS_IdsM_91022]	Definition of imported datatypes of module IdsM
[SWS_IdsM_91024]	Definition of optional interfaces requested by module IdsM
[SWS_IdsM_91030]	Definition of Port IdsMService_{EventName} provided by module IdsM
[SWS_IdsM_91031]	Definition of ImplementationDataType IdsM_SecurityEventIdType
[SWS_IdsM_91033]	Definition of configurable interface IdsM_<ContextDataModifierCallout>
[SWS_IdsM_91034]	Definition of callback function IdsM_CsmNotification
[SWS_IdsM_91035]	Definition of datatype IdsM_TransmissionStateType
[SWS_IdsM_91038]	Definition of API function IdsM_ReportSecurityEvent
[SWS_IdsM_91044]	Definition of ClientServerInterface IdsMService_{EventName}
[SWS_IdsM_91045]	Definition of ClientServerInterface IdsMService_{EventName}
[SWS_IdsM_91046]	Definition of ClientServerInterface IdsMService_{EventName}
[SWS_IdsM_91047]	Definition of ClientServerInterface IdsMService_{EventName}

Table A.6: Changed Specification Items in R25-11

A.2.3 Deleted Specification Items in R25-11

Number	Heading
[SWS_IdsM_00300]	
[SWS_IdsM_00301]	
[SWS_IdsM_00401]	
[SWS_IdsM_00402]	
[SWS_IdsM_00403]	
[SWS_IdsM_00404]	
[SWS_IdsM_00405]	
[SWS_IdsM_00501]	
[SWS_IdsM_00502]	
[SWS_IdsM_01212]	Deprecated Reporting API
[SWS_IdsM_01301]	No additional Interface Option
[SWS_IdsM_01302]	Additional Interface Option: Count
[SWS_IdsM_01303]	Additional Interface Option: Count and timestamp
[SWS_IdsM_02101]	
[SWS_IdsM_02102]	

▽

△

Number	Heading
[SWS_IdsM_02103]	
[SWS_IdsM_02104]	
[SWS_IdsM_02105]	
[SWS_IdsM_02106]	
[SWS_IdsM_02107]	
[SWS_IdsM_02108]	
[SWS_IdsM_02109]	
[SWS_IdsM_91002]	Definition of API function IdsM_SetSecurityEvent
[SWS_IdsM_91003]	Definition of API function IdsM_SetSecurityEventWithContextData
[SWS_IdsM_91014]	Definition of datatype IdsM_TimestampType
[SWS_IdsM_91018]	Definition of API function IdsM_SetSecurityEventWithCount
[SWS_IdsM_91019]	Definition of API function IdsM_SetSecurityEventWithCountContextData
[SWS_IdsM_91020]	Definition of API function IdsM_SetSecurityEventWithTimestampCount
[SWS_IdsM_91021]	Definition of API function IdsM_SetSecurityEventWithTimestampCount ContextData
[SWS_IdsM_91025]	Definition of Port IdsMSmartSensorService_{EventName} provided by module IdsM
[SWS_IdsM_91026]	Definition of Port IdsM_CustomTimestamp required by module IdsM
[SWS_IdsM_91027]	Definition of ClientServerInterface IdsMService_{EventName}
[SWS_IdsM_91028]	Definition of ClientServerInterface IdsMSmartSensorService_{EventName}
[SWS_IdsM_91029]	Definition of ClientServerInterface IdsM_CustomTimestamp
[SWS_IdsM_91032]	Definition of datatype IdsM_ExternalSecurityEventIdType

Table A.7: Deleted Specification Items in R25-11

A.2.4 Added Constraints in R25-11

none

A.2.5 Changed Constraints in R25-11

none

A.2.6 Deleted Constraints in R25-11

none