Document Title	Specification of Intrusion Detection System Protocol
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	981

Document Status	published
Part of AUTOSAR Standard	Foundation
Part of Standard Release	R21-11

Document Change History			
Date	**Release**	**Changed by**	**Description**
2021-11-25	R21-11	AUTOSAR Release Management	• Improved explanations of protocol fields • Increased consistency between overview and detailed tables • Corrections in frame size calculation
2020-11-30	R20-11	AUTOSAR Release Management	• Initial release

Disclaimer

Table of Contents

1 Introduction and overview 5
1.1 Protocol purpose and objectives . . . 5
1.2 Applicability of the protocol . . . 5
1.2.1 Constraints and assumptions . . . 5
1.2.2 Limitations . . . 5
1.3 Dependencies . . . 6
1.3.1 Dependencies to other protocol layers . . . 6
1.3.2 Dependencies to other standards and norms . . . 6
1.3.3 Dependencies to the Application Layer . . . 6
2 Use Cases 7
2.1 UC_0001 "Propagation of Qualified Security Events to the IdsR" . . . 7
3 Protocol Requirements 8
3.1 Requirements Traceability . . . 8
4 Definition of Terms and Acronyms 9
4.1 Acronyms . . . 9
4.2 Abbreviations . . . 10
5 Protocol specification 11
5.1 IDS Message Format . . . 11
5.1.1 IDS Protocol Overview . . . 12
5.1.2 Endianess - Byte Order . . . 13
5.1.3 Independence of communication interface . . . 13
5.1.4 IDS Event Frame . . . 13
5.1.4.1 Protocol Version and Header . . . 14
5.1.4.1.1 Protocol Version . . . 14
5.1.4.1.2 Protocol Header . . . 14
5.1.4.2 IdsM Instance ID and Sensor Instance ID . . . 15
5.1.4.2.1 IdsM Instance ID . . . 15
5.1.4.2.2 Sensor Instance ID . . . 15
5.1.4.3 Event Definition ID . . . 16
5.1.4.4 Count . . . 16
5.1.4.5 Reserved . . . 17
5.1.5 Timestamp . . . 18
5.1.5.1 Timestamp AUTOSAR . . . 19
5.1.5.1.1 Nanoseconds . . . 19
5.1.5.1.2 Seconds . . . 19
5.1.5.2 Timestamp OEM . . . 20
5.1.6 Context Data . . . 21
5.1.6.1 Context Data - Size Long . . . 21
5.1.6.2 Context Data - Size Short . . . 22
5.1.7 Context Data Length Encoding . . . 22

5.1.8 Signature . . . 23
5.1.8.1 Signature Length . . . 23
5.1.8.2 Signature Data . . . 24
5.1.9 IDS Message Separation . . . 25
5.1.9.1 IDS Message Separation Header . . . 26
5.1.9.2 IDS Message Separation Header ID . . . 26
5.1.9.3 IDS Message Separation Header Length . . . 26
5.1.10 PDU Type . . . 27
5.1.11 Example of IDS Messages . . . 27
5.1.11.1 Example IDS Message with Maximum Size . . . 27
5.1.11.2 Example IDS Message with minimum size . . . 28
5.2 Message types . . . 28
5.3 Services / Commands . . . 28
5.4 Sequences (lower layer) . . . 28
5.5 Error messages . . . 29
6 Configuration parameters 30
7 Protocol usage and guidelines 31

1 Introduction and overview

This Protocol Requirements Specification defines the format, message sequences and semantics of the AUTOSAR Protocol Intrusion Detection System (**IDS**).

The document RS IntrusionDetectionSystem [1] describes the elements of a distributed Intrusion Detection System (**IDS**). Please see [1] for an overview of the **IDS** elements.

The **PRS IDS** contributes to the **IDS** by providing the protocol for the transmission of qualified security events (**QSEv**) from an Intrusion Detection System Manager (**IdsM**) instance to an Intrusion Detection System Reporter (**IdsR**) instance.

1.1 Protocol purpose and objectives

As described in [1] **QSEv** can be persisted locally on the **ECU** where the security event was qualified. Alternatively a **QSEv** can be send to the **IdsR**. The **IDS protocol** covers the sending of the **QSEv** from the **IdsM** instance to the **IdsR** instance.

1.2 Applicability of the protocol

The **IDS protocol** supports a push-interface for **QSEv**. The **IdsM** instances push **QSEv** which are configured accordingly to the **IdsR**. A pull interface is not covered by this protocol. It could be realized by storing the QSEv locally in a appropriate component and then accessing the locally stored **QSEv** via regular diagnostic interfaces.

1.2.1 Constraints and assumptions

There are no specific assumptions and constraints for using the **IDS protocol**. It was designed to work for all bus system. The software stack must be able to send and receive **I-PDUs**. The IdsM does not support the reception of QSEvs.

1.2.2 Limitations

There is no limit defined for the context data size. The recommendation is to set the limit for a complete individual **QSEv** to 16 kByte.

1.3 Dependencies

1.3.1 Dependencies to other protocol layers

IdsM instances on the **Classic Platform** use the **PDU Router** to transmit **QSEv** via the **IDS protocol** .

1.3.2 Dependencies to other standards and norms

The elements of the **IDS protocol** can be mapped to the syslog format by the **IdsR** if required for the **SOC**.

1.3.3 Dependencies to the Application Layer

The **IDS protocol** has no dependencies to the application layer. Application layer components can issue security events by using **API** of **IdsM**.

2 Use Cases

The AUTOSAR IDS architecture and functionality is described in [1]. Therefore this chapter is a brief summary of the use case for the protocol.

ID	*Name*	*Description*
0001	Transmission of QSEv	Transmission of qualified security events from IdsM instances to IdsR instance

Table 2.1: Usecases for IDS protocols

2.1 UC_0001 "Propagation of Qualified Security Events to the IdsR"

The main use case for the **IDS protocol** is the propagation of Qualified Security Events **QSEv** to the **IdsR** in a way that is independent from the kind of **ECU** or the used communication mechanism. **IdsM** instances can be allocated to all nodes of the vehicle architecture that are security relevant. This decision is typically based on a security analysis of the vehicle E/E architecture. As a result an **IdsM** instance can be connected to the **IdsR** indirectly via a number of different bus systems as illustrated in Figure 2.1.

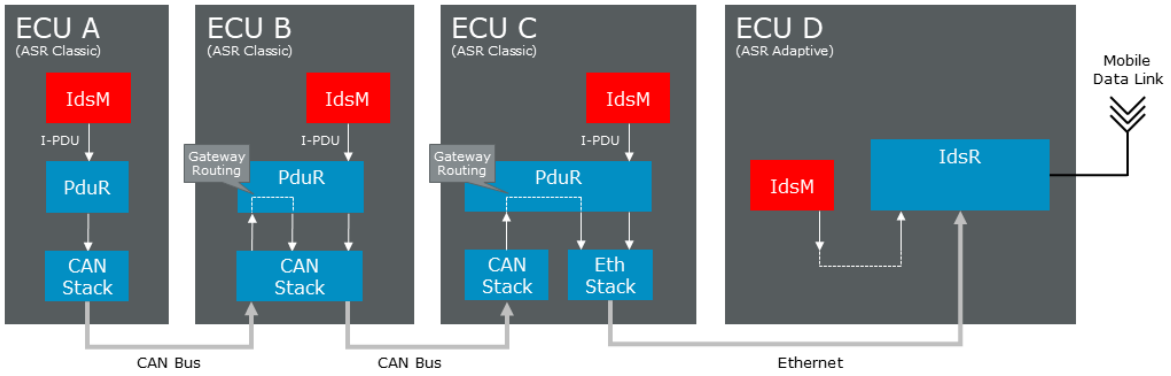

Figure 2.1: Use case for IDS protocol

3 Protocol Requirements

3.1 Requirements Traceability

The following tables reference the requirements specified in the IDS requirement specification [1] and links to the fulfillment of these.

Requirement	Description	Satisfied by
[RS_Ids_00502]	Event Timestamps	[PRS_Ids_00400] [PRS_Ids_00401]
[RS_Ids_00503]	Timestamp Sources	[PRS_Ids_00404]
[RS_Ids_00505]	Authenticity of QSEvs	[PRS_Ids_00600] [PRS_Ids_00601]
[RS_Ids_00510]	The IdsM shall allow to transmit QSEv to the IdsR	[PRS_Ids_00001]
[RS_Ids_00820]	IdsM Security Events	[PRS_Ids_00720]

4 Definition of Terms and Acronyms

4.1 Acronyms

Acronym	Description:
Adaptive Platform	AUTOSAR Adaptive Platform
BSW	Standardized AUTOSAR Software modules, which provides basic functionalities usually required in electronic control unit.
Controller Area Network/Controller Area Network with Flexible Data-Rate	An automotive network communication protocol.
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.
Classic Platform	AUTOSAR Classic Platform
Context Data Buffer	Buffer with variable sizes to fit to the needs of the context data of the SEvs.
ECU	Electronic Control Unit which provides functionalities in electronic system of a car, e.g. brake system or window lifter.
Event Buffer	Buffer to temporarily store the reported SEv.
Event Frame	Main frame of IDS protocol which includes the basic information like the Security Event ID.
Filter Chain	A set of consecutive filters which is applied to security events. The output are Qualified Security Events.
FlexRay	An automotive network communication protocol.
General Purpose I-Pdu	General Purpose Interaction Layer Protocol Data Unit.
Intrusion Detection System	An Intrusion Detection System is a security control which detects and processes security events.
Intrusion Detection System protocol	The IDS protocol specifies the message format which is used by IDS.
Intrusion Detection System 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.
Intrusion Detection System Reporter	The Intrusion Detection System Reporter handles Qualified Security Events received from IdsM instances.
I-PDU Multiplexer	An AUTOSAR Basic Software module which specifies the protocol to multiplex multiple Pdus with one Protocol Control information.
LIN	Local Interconnect Network: serial communication bus to connect sensors and actuators.
Protocol Data Unit Router	An AUTOSAR component responsible for routing of messages independent from underlying communication network.
Protocol Requirement Specification Intrusion Detection System	The specification document which describes all elements of the IDS protocol.
Qualified Security Event (QSEv)	Security events which pass their filter chain are regarded as Qualified Security Events and are sent to the configured sink.
Security Extract	The Security Extract specifies which security events are handled by IdsM instances and their configuration parameters.
Security Events	Onboard security events are reported by BSW, CDD, SWC or other software components or applications to the IdsM.

Acronym	Description:
Security Event Memory	A user defined diagnostic event memory which is independent from the primary diagnostic event memory.
Security Sensors	BSW, CDD, SWC or other software components or applications which report security events to the IdsM.
Security Incident and Event Management	Technology concept to collect, correlate and analyze security incidents to detect a threat.
Sensor	Reporting identity that informs the IdsM module about SEvs. It can be a BSW module, a proprietary CDD or a SWC Application.
Security Operation Centre	Security Operation Center is the Backend of the IDS in which data can be processed and analysed.
Socket Adapter	Socket Adaptor is a Basic Software module of AUTOSAR which creates interface between Pdu-Based communication on service level and socket based TCP/IP

Table 4.1: Acronyms

4.2 Abbreviations

Abbreviation	Description:
AP	AUTOSAR Adaptive Platform
API	Application Programming Interface
BSW	Basic Software
CAN	Controller Area Network
CAN FD	Controller Area Network with Flexible Data-Rate
CDD	Complex Device Driver
CP	AUTOSAR Classic Platform
ECU	Electronic Control Unit
ID	Identifier
IDS	Intrusion Detection System
I-PDU	Interaction Layer Protocol Data Unit
IdsM	Intrusion Detection System Manager
IdsR	Intrusion Detection System Reporter
LIN	Local Interconnect Network
ms	Miliseconds
N-PDU	Network Layer Protocol Data Unit
OEM	Original Equipment Manufacturer
PDU	Protocol Data Unit Router
PRS IDS	Protocol Requirement Specification Intrusion Detection System
QSEv	Qualified Security Event
SecXT	Security Extract
SEv	Security Event
Sem	Security Event Memory
SIEM	Security Incident and Event Management
SOME/IP	Scalable service-Oriented MiddlewarE over IP
SOC	Security Operation Center
SWC	Software Component

Table 4.2: Abbreviations

5 Protocol specification

[PRS_Ids_00001] ⌈The main purpose of the **IDS protocol** is the transmission of qualified security events (**QSEv**) from an Intrusion Detection System Manager (**IdsM**) instance to an Intrusion Detection System Reporter (**IdsR**) instance.⌋*(RS_Ids_00510)*

5.1 IDS Message Format

The IDS protocol is shown in Figure 5.1.

Figure 5.1: IDS Message including Signature

[PRS_Ids_00002] ⌈The **IDS protocol** consists of the standard Event Frame and up to three optional fields. It provides several options to send only minimal data of Qualified Security Event **QSEv** or to extend this data with more details.

Beside the extension with a timestamp or context data, there is also the option to secure the data transport by adding a signature to every **QSEv**. The list below shows examples of configurations and explains the options.
⌋*()*

All options can be configured or switched off independent from each other, so a subset or combination of all options is possible.

1. **[PRS_Ids_00003]** ⌈Standard Qualified Security Event **QSEv** without further data.⌋*()*
2. **[PRS_Ids_00400]** ⌈Qualified Security Event **QSEv** with *Timestamp*: If more precise timestamp is required in addition to the one provided for example by **IdsR**. The sensor or the **IdsM** can add timestamp to every **QSEv**.
   This option must be set by corresponding configuration bit in the protocol header.⌋ *(RS_Ids_00502)* (refer to 5.1.5 *Timestamp*)
3. **[PRS_Ids_00500]** ⌈Qualified Security Event **QSEv** with *Context Data*: The context data includes sensor specific information which are only forwarded to the sink. **IdsM** do not have knowledge on content or structure of this data.
   This option must be set by corresponding configuration bit in the protocol header.⌋ *()* (refer to 5.1.6 *Context Data*)
4. **[PRS_Ids_00600]** ⌈Qualified Security Event **QSEv** with *Signature*: If more secure communication of security events is required a signature can be added to every **QSEv**.
   This option must be set by corresponding configuration bit in the protocol header.⌋ *(RS_Ids_00505)* (refer to 5.1.8 *Signature*)

5.1.1 IDS Protocol Overview

In figure Figure 5.2 you can find an overview on all elements of the **IDS protocol**.

FieldName	Length	Description of the data
Protocol Version	4 Bit	The version of the IdsM protocol
Protocol Header	4 Bit	IdsM protocol header information: Bit[0]: 0 - No Context Data included, 1 - Context Data included Bit[1]: 0 - No Timestamp included, 1 - Timestamp included Bit[2]: 0 - No Signature included, 1 - Signature included Bit[3]: reserved
IdsM Instance Id	10 Bit	Unique identifier of the sending IdsMinstance 0-1023
Module Instance Id	6 Bit	Identifier to differ between multiple instances of modules
Event Id	16 Bit	Unique identifier of a Security Event: Range of AUTOSAR internal IDs: 0…0x7FFF Range of Customer specific IDs: 0x8000…0xFFFF
Count	16 Bit	Number of IdsM calls which result in the current event after processing the configured filter, e.g. *EventAggregation*
Reserve	8 Bit	Reserved for future use
Timestamp	8 Bytes	Timestamp / Tickstamp when event was detected: (optional) Byte[0] Bit[7]=0: AUTOSAR Standard, Byte[0] Bit[6]: reserved Byte[0] Bit[7]=1: OEM Specific / Custom Timestamp Resolution in ms. Maybe not necessary for every event type. If not set, field is filled by IdsR. If not authentic time, IdsR might recalculate the time and insert a new value
Context Data Length	1 or 4 Bytes	Length information of Context Data. Only available if Context Data exists. (option) Most Significant Bit of first byte Context Data signals if Context Data Length is encoded in 7 Bit or 31 Bit: Context Data Byte[0] Bit[7]=0: Length is encoded in 7 Bits - Byte[0] Bit[0..6] - Valid values: 1..127 Bytes Context Data Byte[0] Bit[7]=1: Length is encoded in 31 Bits - Byte[0] Bit[0..6], Byte[1..3] Bit[0..7] - Valid values: 1..(2^31) - 1 Bytes
Context Data	1…(2^31) -1 Bytes	Binary blob attached by the sensor: (optional)
Signature Length	2 Bytes	Length information of Signature. Only available if Signature exists. (optional) Signature Byte[0..1]: Signature Length 1..65535 Bytes
Signature	1..65535 Bytes	Signature for authentification of security event: (optional) Signature calculated with Eventframe + Optional Timestamp + Optional Context Data Signature Byte[2..n]: Signature Data - configurable via MetaModel

Figure 5.2: Intrusion Detection System Protocol Overview

5.1.2 Endianess - Byte Order

[PRS_Ids_00004] ⌈The **IDS protocol** uses big endianess as byte order also known as Motorola format. This is equal to the network byte order, e.g. used by ethernet. In the tables and descriptions of this section, the byte numbers increase in the same sequence as the bytes are transmitted in the **IDS** message, starting from 0.

The first byte is the **M**ost **S**ignifcant **B**yte (MSB), usually Byte 0
the last byte is the **L**east **S**ignificant **B**yte (LSB).
The bit numbers decrease, the **m**ost **s**ignificant **b**it (msb) of a byte being bit 7 and the **l**east **s**ignificant **b**it (lsb) 0.⌋*()*

5.1.3 Independence of communication interface

[PRS_Ids_00005] ⌈The **IDS protocol** is independent from the used hardware and the underlying communication interface (e.g. CAN, Ethernet, FlexRay). It is optimised to fit to standard CAN bus communication with the minimum required information on **security event**. Also ethernet communication is applicable.⌋*()*

5.1.4 IDS Event Frame

Figure 5.3 shows the **Event Frame** of **IDS protocol**.

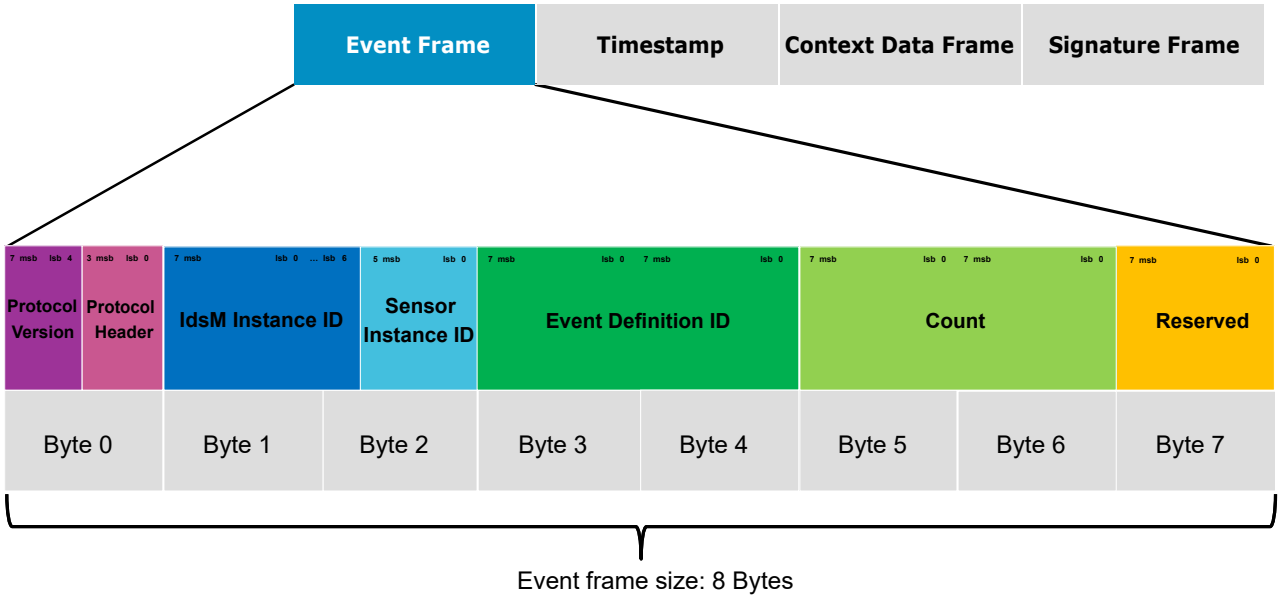

Figure 5.3: IDS Event Frame

[PRS_Ids_00006] ⌈The **IDS Event Frame** consists of 8 Byte as detailed above.⌋*()*

5.1.4.1 Protocol Version and Header

Byte 0							
Bit 7	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
Protocol Version				**Protocol Header**			
				Re-served	Signa-ture	Times-tamp	Context Data

Table 5.1: Layout of Protocol Version and Header

5.1.4.1.1 Protocol Version

[PRS_Ids_00008] ⌈The version information of the IDS protocol:

- Bit[7..4] : 0-15

Formula for calculation:
ProtocolVersion = (BYTE0 & 0xF0) >> 4
The used version number for this specification of the IDS protocol shall be 1.⌋*()*

5.1.4.1.2 Protocol Header

[PRS_Ids_00009] ⌈IDS protocol header information, includes configuration bits to switch specific functionalities on or off:

- Bit[0]: Context Data included

  0: No Context Data included

  1: Context Data included
- Bit[1]: Timestamp included

  0: No Timestamp included

  1: Timestamp included
- Bit[2]: Signature included

  0: No Signature included

  1: Signature included
- Bit[3]: reserved

Formula for calculation:
ProtocolHeader = (BYTE0 & 0x0F)⌋*()*

Note:
[PRS_Ids_00010] ⌈Only if Timestamp, Context Data or Signature is available, the corresponding Protocol Header Bit is set to 1.
Context Data or Signature will never be transmitted with Length=0.⌋*()*

[PRS_Ids_00011] ⌈Reserved Bits should be preset with value 0. On receiver side those bits should be ignored.⌋*()*

5.1.4.2 IdsM Instance ID and Sensor Instance ID

[PRS_Ids_00012] ⌈Table 5.2 shows the combined element IdsM and Sensor Instance ID.⌋*()*

Byte 1								Byte 2							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
IDS Instance ID								Sensor Instance ID							

Table 5.2: IdsM Instance ID, Sensor Instance ID

5.1.4.2.1 IdsM Instance ID

[PRS_Ids_00013] ⌈Unique identifier of the **IdsM** instance which sends the **security event**.
IdsM Instance ID range: 0-1023.
Usually there is one **IdsM** instance in one **ECU**. In case of complex **ECU** with Classic and Adaptive components, e.g. Multi Controller or Multi Partition devices it is possible that there are multiple **IdsM**. One **IdsM** in **Classic Platform** and one **IdsM** in **Adaptive Platform**. In such a constellation both **IdsM** must be configured with different IDS Instance ID.
Formula for calculation:
IdsM Instance ID (10 Bits) = ((BYTE2 & 0xC0) >> 6) |((BYTE1 << 2))⌋*()*

5.1.4.2.2 Sensor Instance ID

[PRS_Ids_00014] ⌈Identifier to differentiate between multiple instances of same kind of sensor module.
Sensor Instance ID range: 0-63
e.g. Multiple CanDrv in one **ECU** can issue "same" **security event**. To differentiate these the Sensor Instance ID is used.
In case there is only one instance of the sensor in the configuration, the value of the Sensor Instance ID shall be, by default, set to 0.
Note:

The Sensor Instance ID shall be set at configuration of the corresponding **Basic Software module**.

Formula for calculation:
Sensor Instance ID (6 Bits) = (BYTE2 & 0x3F)⌋()

5.1.4.3 Event Definition ID

The Event Definition ID is shown in Table 5.3.

Byte 3	Byte 4
Event Definition ID	

Table 5.3: Event Definition ID

[PRS_Ids_00015] ⌈The Event Definition ID is a unique identifier of a **security event**. It describes the kind of a **security event**.⌋()

[PRS_Ids_00016] ⌈If a sensor generates multiple **security events** of same kind it is called Event instance.⌋()

[PRS_Ids_00017] ⌈The range for the Event Definition ID is split into three scopes:

1. AUTOSAR internal IDs: 0-0x7FFF (max. 32768 security events)
2. Customer specific IDs: 0x8000-0xFFFE (max. 32767 security events)
3. Invalid ID: 0xFFFF

⌋()

5.1.4.4 Count

Table 5.4 shows the IDS element Count.

Byte 5	Byte 6
Count	

Table 5.4: Count

[PRS_Ids_00018] ⌈The count represents the number of **IdsM API** calls which result in the current **Qualified Security Event**. When an event is created, its count is initialized to 1. However, filters like Event Aggregation may combine several events into a single one. The count of this event is set to the sum of the counts of all aggregated events. If the **security event** is send by a smart sensor which already filters and preset the count value, this preset is just added to the count of **IdsM**. So the final count is the sum of the count of the sensor and the result of **IdsM** processing.⌋()

5.1.4.5 Reserved

The Reserved byte is shown in Table 5.5.

Byte 7
Reserved

Table 5.5: Reserved

[PRS_Ids_00019] ⌈The Byte[7] of the **Event Frame** of **IDS protocol** is reserved for future use.⌋*()*

Note:
[PRS_Ids_00020] ⌈Reserved Bytes should be preset with value 0. On receiver side those bytes should be ignored.⌋*()*

5.1.5 Timestamp

Details on timestamp are shown in Figure 5.4.

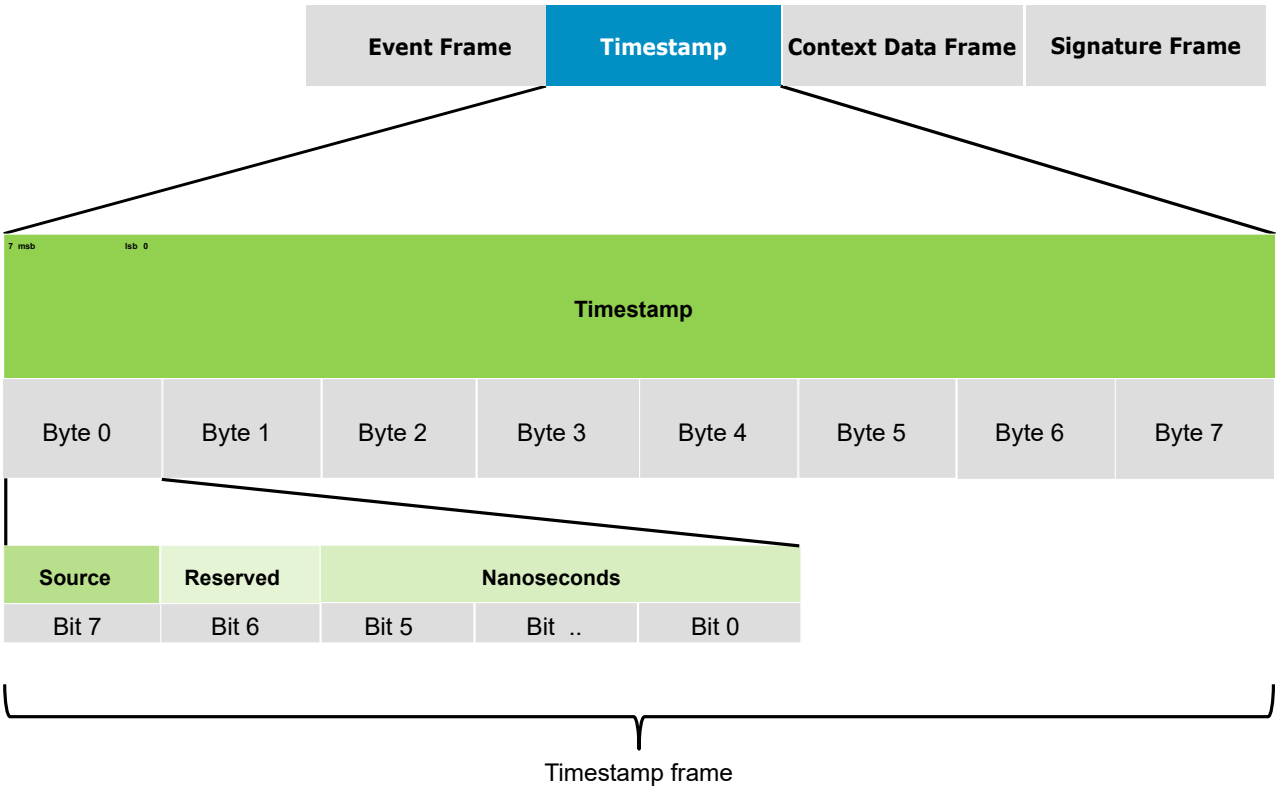

Figure 5.4: Timestamp

[PRS_Ids_00401] ⌈The **IDS Protocol** provides Timestamp as a configurable option.⌋*(RS_Ids_00502)*

[PRS_Ids_00402] ⌈It is logged when **security event** was detected the first time (first occurence).⌋*()*

[PRS_Ids_00403] ⌈Resolution in ms is required. The Timestamp shall be encoded with 64 Bits in total to fit into a single CAN frame.⌋*()*

[PRS_Ids_00404] ⌈Different sources for Timestamp can be configured in the **IDS Protocol**.

- Bit[7]: Timestamp source

  0: AUTOSAR Standard CP: StbM - AP: ara::tsync

  1: Auxiliary / OEM Specific timestamp

- Bit[6]: reserved

⌋*(RS_Ids_00503)*

5.1.5.1 Timestamp AUTOSAR

Timestamp				
Byte 0			Byte 1	Byte 2	Byte 3
Bit 7	Bit 6	Bit 5 .. 0			
Source	Reserved	Nanoseconds			

Table 5.6: Timestamp Source and Nanoseconds

Timestamp			
Byte 4	Byte 5	Byte 6	Byte 7
Seconds			

Table 5.7: Timestamp Seconds

[PRS_Ids_00405] ⌈For the **IDS Protocol** AUTOSAR time format combines the timestamps for nanoseconds with 30 Bits and seconds with 32 Bits.⌋*()*

5.1.5.1.1 Nanoseconds

[PRS_Ids_00406] ⌈For nanoseconds only 30 Bits are required to encode 0..999 999 999 ns = 10^{-9} seconds.⌋*()*

Note:
AUTOSAR Time Synchronisation Protocol (e.g. stbm in **CP**) uses 32 Bits for nanoseconds. The truncation of nanoseconds for **IDS Protocol** does not limit the resolution of the timestamp.

5.1.5.1.2 Seconds

[PRS_Ids_00407] ⌈Seconds are encoded with 32 Bits which result in approximately 127 years resolution.⌋*()*

Note:
For details please refer to Time Synchronisation Protocol SWS-TimeSynchronisation [2]

5.1.5.2 Timestamp OEM

Timestamp				
Byte 0		Byte 1	Byte 2	Byte 3
Bit 7	Bit 6 .. 0			
Source	OEM Timestamp			

Table 5.8: OEM Timestamp format

[PRS_Ids_00408] ⌈OEM time source offers the option to use other time protocol. The length is limited to 63 Bits. An interface to OEM application is required. Accuracy is defined by OEM.⌋*()*

5.1.6 Context Data

[PRS_Ids_00501] ⌈The **IDS protocol** provides an optional feature to enrich the standard security event transfered in the **Event Frame** with more detailed information. Therefore context data can be added. It is a binary blob attached by the **sensor**. These data includes specific detailed information about the security event which can be used by the **SOC** for improved analysis of the security incident, e.g. a malformed message detected by a communication sensor.
IdsM has no knowledge of the content or structure of these data. Only the issuing **sensor** and the **Backend** or **SOC** knows it.⌋*()*

There are two variants of context data with different sizes:

5.1.6.1 Context Data - Size Long

Figure 5.5 shows the "Context Data Size Long" with 4 Bytes length field.

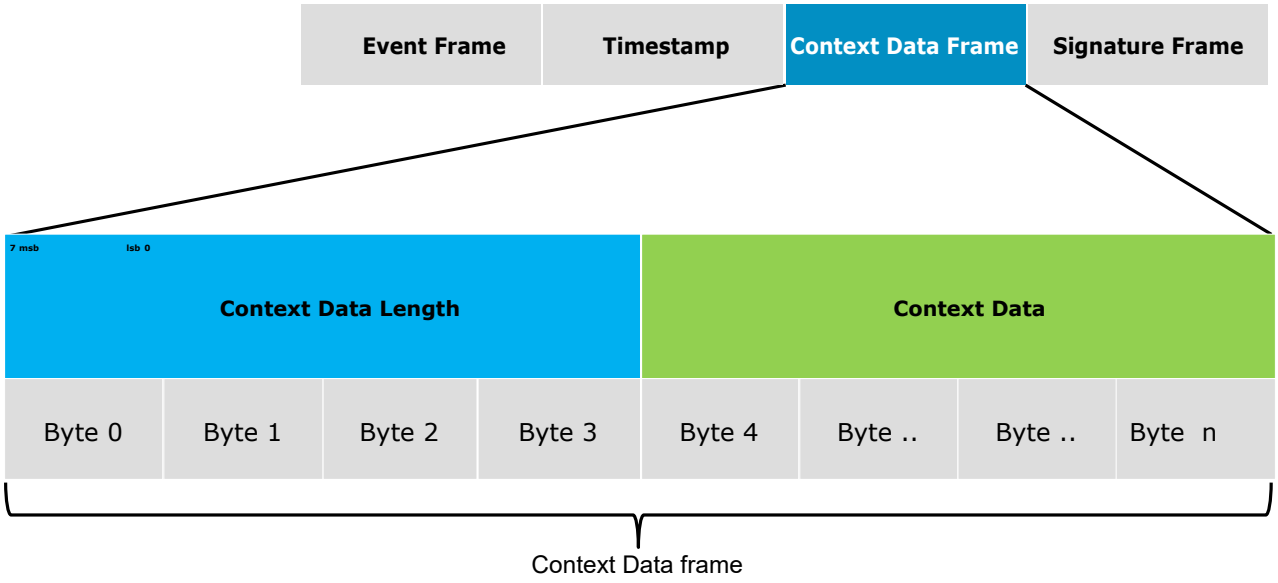

Figure 5.5: Context Data Size Long

[PRS_Ids_00502] ⌈The "Context Data Size Long" includes a 4 Bytes length field. Up to 2^{31}-1 context data bytes can be transmitted.⌋*()*

5.1.6.2 Context Data - Size Short

In Figure 5.6 the alternative version "Context Data Size Short" is shown.

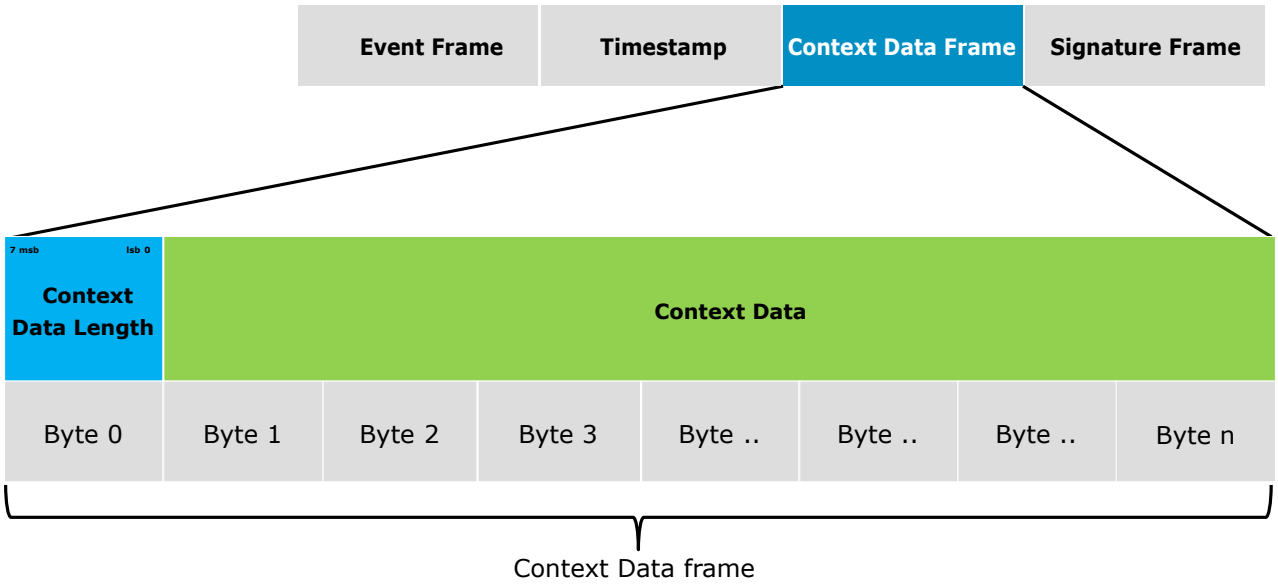

Figure 5.6: Context Data Size Short

[PRS_Ids_00503] ⌈The "Context Data Size Short" is the alternative version with 1 Byte length field for max. 127 Bytes context data.⌋*()*

5.1.7 Context Data Length Encoding

Context Data							
Byte 0							
Bit 7	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
Length Format	Context Data Length						

Table 5.9: Context Data

[PRS_Ids_00504] ⌈In Table 5.9 the encoding of the Context Data Length is shown.

- Context Data Byte[0] Bit[7]

  0: 7 Bits length information encoded in Context Data Byte[0] Bit[0..6]: 1-127 Bytes

  1: 31 Bits Length Information encoded in Context Data Byte[0..3] Bit[0..30]: 1..(2^{31}-1) Bytes

Most Significant Bit (msb) of first byte Context Data (MSB) signals if the length is encoded in 7 Bits (1 Byte) or 31 Bits (4 Bytes).⌋*()*

5.1.8 Signature

[PRS_Ids_00601] ⌈The **IDS protocol** provides an optional feature to make the transmission of **QSEv** more secure. A digital signature can be added to the **IDS message**. It can be used to ensure authenticity as well as to prove integrity of signed messages from the **IdsM** via all communication systems until reaching the **Backend** or **SOC** (End2End-Security).⌋*(RS_Ids_00505)*

Figure 5.7 shows the signature option of the **IDS protocol**.

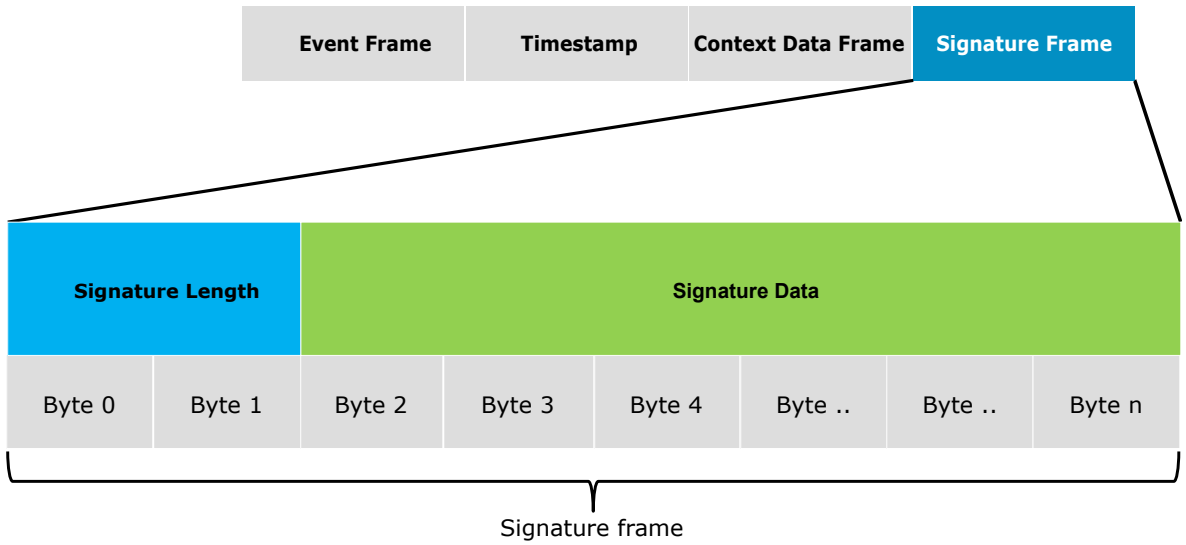

Figure 5.7: Signature

5.1.8.1 Signature Length

[PRS_Ids_00602] ⌈

- Signature Length is encoded in 2 Bytes:
  Signature Length Byte[0..1]: Signature Length 1..65535

⌋*()*

5.1.8.2 Signature Data

Signature							
Byte 2	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 5**	**Byte ..**	**Byte ..**	**Byte n**
Signature Data							

Table 5.10: Signature

Table 5.10 shows the signature data. **[PRS_Ids_00603]** ⌈

- Signature Data Byte[2..65537]: Signature data

The cryptographic value of the Signature of **`security event`** is calculated with the serialized data of:
Event Frame + optional Timestamp + optional Context Data.
Which kind of cryptoalgorithm is used, depends on the system.
The **`IDS protocol`** does not prescribe any specific algorithm nor the format.⌋*()*
(also refer to 5.1.4 Event Frame, 5.1.5 Timestamp and 5.1.6 Context Data.)

5.1.9 IDS Message Separation

[PRS_Ids_00800] ⌈On ethernet the IDS Message Separation Header is mandatory. It is used to address **IDS messages** unambiguously. In addition to the transmission of a single **IDS message** via ethernet, multiple **IDS messages** can be collected and sent within a single ethernet frame.⌋*()*

[PRS_Ids_00801] ⌈An unique ethernet port address should be used for **IDS** communication.⌋*()*

[PRS_Ids_00802] ⌈**SOME/IP** and **IDS** messages should not be mixed on same port as they can't be distinguished properly by the receiver.⌋*()*

Note:
The **IdsR** typically is connected via Ethernet. But as already mentioned also other automotive communication buses and protocols are supported. Regarding message separation header the following should be considered:

- **CAN FD**: The I-Pdu-Multiplexer [3] supports collecting of multiple **IDS messages** within one message. Because of the size restrictions on **CAN** the I-Pdu-Multiplexer typically uses short header or no header option. Therefore the IDS Message Separation Header is normally not used on **CAN** buses.
- **FlexRay**: The PDU Packing feature supports collecting of multiple **IDS messages** within one message. It does not use separation headers but update bits to identify available parts.
  For more details please refer to SWS FlexRay Interface [4].
- **CAN** (Standard): is transferred without IDS Message Separation Header.
- **LIN**: is transferred without IDS Message Separation Header.

5.1.9.1 IDS Message Separation Header

Figure 5.8 shows the IDS Message Separation Header.

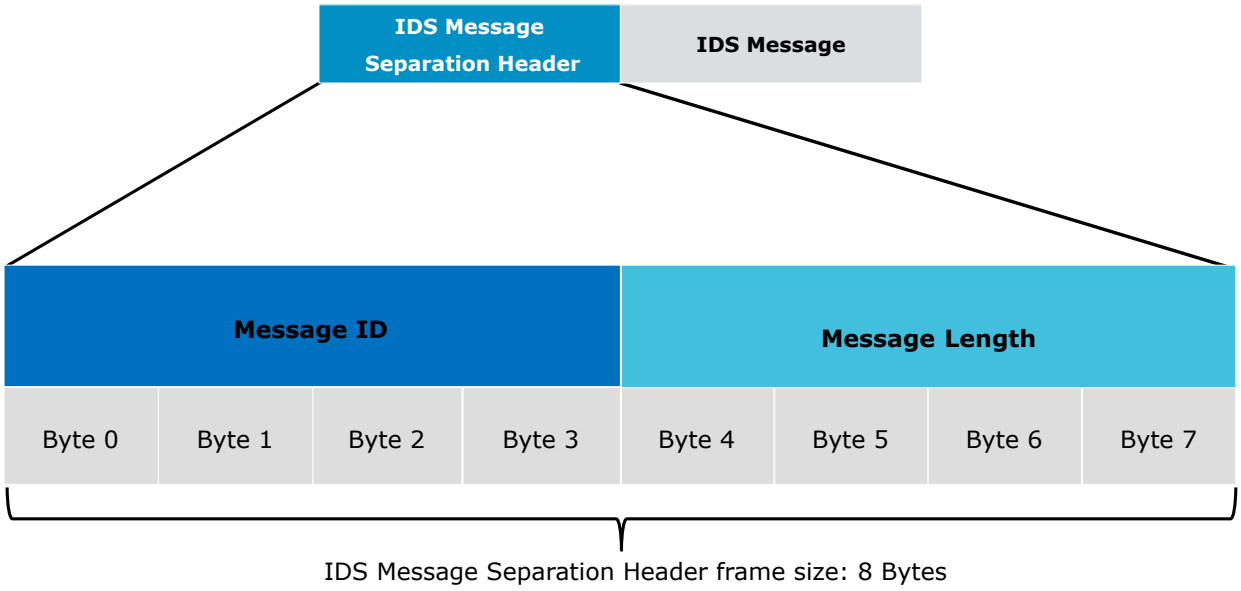

Figure 5.8: IDS Message Separation Header

[PRS_Ids_00803] ⌈
The IDS Message Separation Header consists of a 4 byte **ID** field for unique identification at the receiver and a 4 byte length field specifying the data length. Both in big endian byte order.⌋ *()*

5.1.9.2 IDS Message Separation Header ID

[PRS_Ids_00804] ⌈The IDS Message Separation Header ID is encoded in 4 byte. It is an arbitrary number, preferable 0.⌋ *()*

Note:
In AUTOSAR **CP** the IDS Message Separation Header ID shall be set at configuration of the Socket Adapter and I-PDU Multiplexer.
For details please refer to SWS-Socket Adaptor [5] and SWS-IPDUMultiplexer [3].

5.1.9.3 IDS Message Separation Header Length

[PRS_Ids_00805] ⌈The IDS Message Separation Header Length is calculated by adding IDS Message length and static IDS Message Separation Header Length (8 Bytes). It is encoded in 4 bytes.
The possible range is:

Message Length Byte[0..3]: 16.. 2.147.549.212 Bytes.⌋()
The minimum length is 16 Bytes: IDS Message Separation Header (8 Bytes) plus minimal **IDS message** which is the **Event Frame** (8 Bytes) without any options configured.
Please also refer to 5.1.11.2 Example IDS Message with Minimum Size.

The maximum length depends on the configured options.
In case all options are configured with maximum size and the IDS Message Separation Header is used the totale size message is 2.147.549.212 Bytes.
For details please refer to 5.1.11.1 Example IDS Message with Maximum Size.

Note:
AUTOSAR platforms:

- **CP**: The IDS Message Separation Header corresponds to the **N-PDU** mechanism which is supported by SocketAdaptor/I-PDU-Multiplexer [5] / [3].
- **AP**: The IDS Message Separation Header must be generated by **IdsM**.

5.1.10 PDU Type

Note:
In the **CP IDS protocol** uses **GeneralPurposeIPdu** (Interaction Layer Protocol Data Unit) of type **IDS** for transmission of Qualified Security Event **QSEv**.
For details refer to System Template [6], Chapter Communication.

5.1.11 Example of IDS Messages

5.1.11.1 Example IDS Message with Maximum Size

[PRS_Ids_00900] ⌈All options of IDS protocol configured with maximum size:

- Option Timestamp AUTOSAR is configured.
- Option Context Data Size Long is configured.
- Option Signature is configured.

Event Frame: 8 Bytes
Timestamp: 8 Bytes
Context Data Size Long: 2^{31}-1 Bytes = 2.147.483.647 Bytes
Context Data Size Long Length Encoding: 4 Bytes
Signature: 65535 Bytes
Signature Length Encoding: 2 Bytes
IDS Message = 8 + 8 + 2.147.483.647 + 4 + 65535 + 2 = 2.147.549.204 Bytes

For CAN Bus:
Maximum message size with CAN TP = 2^{32} -1 = 4.294.967.295

For Ethernet:
IDS Message Separation Header must be added with 8 Bytes:
Maximum IDS Message with IDS Separation Header:
8 Bytes + 2.147.549.204 Bytes = 2.147.549.212 Bytes

Maximum Size which can be encoded with 4 Bytes for IDS Message Separation Header:
4 Bytes = 2^{32} = 4.294.967.296

This ensures that IDS messages with maximum size can be transfered via the standard automotive bus system!⌋*()*

5.1.11.2 Example IDS Message with minimum size

[PRS_Ids_00901] ⌈No option of IDS protocol is configured - minimal size:

Event Frame: 8 Bytes
IDS Message = 8 Bytes⌋*()*

5.2 Message types

Currently not used for **IDS Protocol**.

5.3 Services / Commands

Currently not used for **IDS Protocol**.

5.4 Sequences (lower layer)

Currently not used for **IDS Protocol**.

5.5 Error messages

IDS protocol do not send specific error messages.

[PRS_Ids_00720] ⌈In case of internal errors of the IdsM specific **Qualified Security Events** are send to the configured sinks:

1. Security Event Buffer overflow: There are no more **event buffers** available to process the event.
2. Context Data Buffer overflow: There are no more **context data buffers** available to store the context data.
3. Traffic Limitation overflow: The current traffic exceeds a configured limit.

⌋*(RS_Ids_00820)*
Note:
The IDs for these events are derived from **Security Extract** (**SecXT**). For details refer to Security Extract Template [7].

6 Configuration parameters

Currently not used for **IDS Protocol**.

7 Protocol usage and guidelines

Currently not used for **IDS Protocol**.

References

[1] Requirements on Intrusion Detection System
AUTOSAR_RS_IntrusionDetectionSystem

[2] Specification of Time Synchronization
AUTOSAR_SWS_TimeSynchronization

[3] Specification of I-PDU Multiplexer
AUTOSAR_SWS_IPDUMultiplexer

[4] Specification of FlexRay Interface
AUTOSAR_SWS_FlexRayInterface

[5] Specification of Socket Adaptor
AUTOSAR_SWS_SocketAdaptor

[6] System Template
AUTOSAR_TPS_SystemTemplate

[7] Security Extract Template
AUTOSAR_TPS_SecurityExtractTemplate