Elektrobit

EB tresos®Safety E2E Transformers safety manual

EB tresos ®Safety E2E Transformers

Date: 2020-07-29, ID: EBASCE2ESE-519, Document version 1.6, Status: RELEASED

Elektrobit Automotive GmbH
Am Wolfsmantel 46
91058 Erlangen, Germany
Phone: +49 9131 7701 0
Fax: +49 9131 7701 6333
Email: info.automotive@elektrobit.com

Technical support

https://www.elektrobit.com/support

Legal disclaimer

Confidential information.

ALL RIGHTS RESERVED. No part of this publication may be copied in any form, by photocopy, microfilm, retrieval system, or by any other means now known or hereafter invented without the prior written permission of Elektrobit Automotive GmbH.

All brand names, trademarks, and registered trademarks are property of their rightful owners and are used only for description.
Copyright 2022, Elektrobit Automotive GmbH.

Table of Contents

1. Document history 6
2. Document information 8
2.1. Objective 8
2.2. Scope and audience 8
2.3. Quality and safety statement 8
2.4. Motivation 9
2.5. Structure 9
2.6. Terminology 9
2.6.1. Keywords 9
2.6.2. Signal Words 10
3. About the Safety Transformers 11
3.1. Architecture of the surrounding system 11
3.1.1. Technical overview 11
3.1.2. Failure model of E2E communication 12
3.1.2.1. Term (object) definitions 12
3.1.2.2. Process definitions 14
3.1.2.3. E2E communication link failure modes 15
3.1.3. Protected communication link 16
3.1.4. Safety Transformers in AUTOSAR 17
3.2. Description of Safety Transformers 19
3.2.1. Identification of the Safety Transformers 19
3.2.2. Functional requirements of Safety Transformers 19
3.2.3. Assumed safety requirements of Safety Transformers 19
3.2.3.1. Top level requirements for the Safety Transformers 20
3.2.3.1.1. EB_E2ESE000070: Freedom from interference of the data element exchange (S) 20
3.2.3.2. Claims for the Safety Transformers 21
3.2.3.2.1. EB_E2ESE000071: Protection of data element exchange (S) 21
3.2.3.2.2. EB_E2ESE000072: Communication failure detection (S) 22
3.2.3.2.3. EB_E2ESE000073: Configurable tolerance level (S) 23
3.2.3.2.4. EB_E2ESE000074: Source-specific error indication (S) 24
3.2.3.2.5. EB_E2ESE000100: Efficient communication of large data elements (S) 25
3.2.4. Safety mechanism used by Safety Transformers 25
3.2.5. System context of the EB tresos Safety E2E Transformers 26
3.2.6. Structure of the Safety Transformers BSW 27
3.2.7. Dynamic behavior of the Safety Transformers 29
3.2.7.1. Interaction view 29
3.2.7.1.1. Initialization of Safety Transformers 29

3.2.7.1.2. Use case: ComXf ... 30
3.2.7.1.2.1. ComXf seen in context ... 30
3.2.7.1.2.2. ComXf seen in composition ... 32
3.2.7.1.3. Use case: SomeIpXf ... 33
3.2.7.1.3.1. SomeIpXf seen in context ... 33
3.2.7.1.3.2. SomeIpXf seen in composition ... 34
3.2.8. Robustness of Safety Transformers ... 35
3.2.8.1. Robustness against hardware faults ... 35
3.2.8.2. Robustness against systematic software errors ... 35
3.2.8.3. Robustness against configuration errors ... 35
3.2.8.4. Robustness against resource conflicts ... 36
3.2.8.5. Robustness against interrupt overload ... 36
3.2.8.6. Robustness against input errors ... 36
3.2.8.7. Robustness against data starvation ... 36
3.2.9. What Safety Transformers does not does ... 36
3.2.10. Backward compatibility ... 36
3.2.11. Change control ... 36
3.2.12. Limitations of Safety Transformers ... 37
4. Using the Safety Transformers ... 38
4.1. Prerequisites ... 38
4.2. Installing Safety Transformers ... 38
4.3. Verifying the integrity of the Safety Transformers sources ... 39
4.4. Field monitoring ... 40
4.5. Correct use and integration of the Safety Transformers ... 40
4.5.1. Tooling workflow of the Safety Transformers ... 40
5. Safety element out of context (SEooC) definition ... 44
5.1. Functional scope of the SEooC ... 44
5.1.1. Provided functionality ... 44
5.1.2. Assumed employment ... 44
5.1.3. Assumed external functionality ... 44
6. Configuration verification criteria ... 45
6.1. Configuration Rules ... 45
6.1.1. Workflow step: Configuring ... 45
6.1.2. Workflow step: Generating ... 46
6.1.3. Workflow step: Verifying generated files ... 46
6.2. Assumptions of Safety Transformers ... 47
6.2.1. Verifying the Software ... 47
6.2.1.1. Generic assumptions ... 47
6.2.1.2. Verifying the Rte ... 48
6.2.2. Verifying the hardware ... 49
6.2.3. Verifying the software tools ... 50
6.2.4. Verifying the communication description ... 50

6.2.5. Verifying the application 51
6.2.6. System assumptions 52
A. Safety Checklist 53
B. Safety Checker 56
B.1. Safety Checker Usage 56
B.2. Safety Check Report 57
C. Document configuration information 58
Glossary 59
Bibliography 62

1. Document history

Version	Date	Author	State	Remarks
0.1	2016-06-09	Elektrobit Automotive GmbH	DRAFT	Initial version of safety manual
0.2	2016-10-25	Elektrobit Automotive GmbH	DRAFT	Update structure of the document
0.3	2016-11-02	Elektrobit Automotive GmbH	DRAFT	Update due to review findings
0.4	2017-05-20	Elektrobit Automotive GmbH	DRAFT	Generic update
0.5	2017-11-02	Elektrobit Automotive GmbH	DRAFT	Update structure of the document, refine application assumptions
0.6	2017-12-07	Elektrobit Automotive GmbH	PROPOSED	Refine configuration assumptions and robustness, adding safety check list
0.7	2018-01-24	Elektrobit Automotive GmbH	PROPOSED	Update due to inspection findings
0.8	2018-02-02	Elektrobit Automotive GmbH	PROPOSED	Added Checker tool section
0.9	2018-02-13	Elektrobit Automotive GmbH	PROPOSED	Added SomeIpXf transformer configuration rules, update verification of delivery content via SHA-1 hashes
1.0	2018-02-14	Elektrobit Automotive GmbH	RELEASED	Set to Released: ASCE2ESE-693
1.1	2018-10-30	Elektrobit Automotive GmbH	PROPOSED	Updated Checker tool installation, added example for failing integrity check, adding EB_E2ESE061030, update user documentation references

Version	Date	Author	State	Remarks
1.2	2019-03-11	Elektrobit Automotive GmbH	PROPOSED	Updated list of generated files to use in chapter Safety Checker Usage, added sub-clause for mixed configurations
1.3	2019-04-03	Elektrobit Automotive GmbH	PROPOSED	Updated chapter 6.2.1. Verifying the Software
1.4	2019-07-17	Elektrobit Automotive GmbH	RELEASED	Update EB_E2ESE061029, Update B.1. Safety Checker Usage
1.4	2019-07-17	Elektrobit Automotive GmbH	RELEASED	Set to Released: ASCE2ESE-825
1.5	2020-03-04	Elektrobit Automotive GmbH	RELEASED	Adding generic assumption EB_E2ESE061031
1.6	2020-07-29	Elektrobit Automotive GmbH	RELEASED	Updated Appendix A. Safety Checklist #2

Table 1.1. Document history

2. Document information

2.1. Objective

The objective of this document is to provide you with all the information necessary to ensure that *EB tresos Safety E2E Transformers* (short Safety Transformers) is used in a safe way in your project.

The goal of the safety manual is to provide the following information:

- Definition of the *Safety Transformers*.
- Information and requirements for the integration of the *Safety Transformers* into an item.
- Information and requirements for the use of the *Safety Transformers* in an item.
- Requirements from the *Safety Transformers* on the item.
- Information about the customer interface of the *Safety Transformers*.

2.2. Scope and audience

This safety manual describes the use of *Safety Transformers* in system applications that have safety requirements up to ASIL-D. It is valid for all projects and organizations that use *Safety Transformers* in a safety-related environment. *Safety Transformers* is intended to be used in AUTOSAR ECU projects.

The intended audience of this document is:

Professionals in embedded automotive systems with the professional qualification in the area of functional safety, communication networks, and AUTOSAR.

2.3. Quality and safety statement

Information about the quality level and safety status of *Safety Transformers* release is provided in the quality statement. If such a statement is not available the software shall be considered as prototype level and must not be used in mass production projects.

2.4. Motivation

This manual provides information on how to correctly use *Safety Transformers* in safety-related projects. If *Safety Transformers* is used differently, *Safety Transformers* code might not comply with the assumed safety requirements, which are defined in Section 3.2.3, “Assumed safety requirements of Safety Transformers”.

2.5. Structure

- Chapter 2, “Document information“: (this chapter) provides a brief introduction of this document and its structure.
- Chapter 3, “About the Safety Transformers“: introduces Safety Transformers and the environment in which it can be used, safety requirements, assumptions, and limitations.
- Chapter 4, “Using the Safety Transformers“: describes how to correctly use Safety Transformers.
- Chapter 5, “Safety element out of context (SEooC) definition“: describes the Safety Element out of Context (SEooC) for Safety Transformers.
- Chapter 6, “Configuration verification criteria“: describes configuration constraints and assumption of the Safety Transformers.
- Appendix A, “Safety Checklist“: describes all required activities that need to be performed for a safe use of Safety Transformers.
- Appendix B, “Safety Checker“: describes the use of the *Safety Transformers Checker*.

2.6. Terminology

All technical terms, definitions and abbreviated terms given in INTERNATIONAL STANDARD ISO 26262-1 [ISO26262-1_1ST] apply.

In addition, the following tables list the interpretation of specific key words and signal words which shall be used for the requirements.

2.6.1. Keywords

Term	Description
MUST	This word, or the terms "REQUIRED" or "SHALL", mean that the definition is an absolute requirement of the specification. [RFC_2119]

Term	Description
MUST NOT	This phrase, or the phrase "SHALL NOT", mean that the definition is an absolute prohibition of the specification. [RFC_2119]
SHOULD	This word, or the adjective "RECOMMENDED", mean that there may exist valid reasons in particular circumstances to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different course. [RFC_2119]
SHOULD NOT	This phrase, or the phrase "NOT RECOMMENDED" mean that there may exist valid reasons in particular circumstances when the particular behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label. [RFC_2119]
MAY	This word, or the adjective „OPTIONAL“, means that an item is truly optional. One vendor may choose to include the item because a particular marketplace requires it or because the vendor feels that it enhances the product while another vendor may omit the same item. An implementation, which does not include a particular option, MUST be prepared to interoperate with another implementation, which does include the option, though perhaps with reduced functionality. In the same vein an implementation, which does include a particular option, MUST be prepared to interoperate with another implementation, which does not include the option (except, of course, for the feature the option provides.) [RFC_2119]

2.6.2. Signal Words

Term	Description
IF	The word *if* defines that an action is only performed in case the specified logical condition evaluates to true.

3. About the Safety Transformers

The *Safety Transformers* facilitates the exchange of safety-related data between AUTOSAR SWCs by protecting the data exchange against the effects of faults along the communication path, including random hardware faults and systematic software faults.

3.1. Architecture of the surrounding system

3.1.1. Technical overview

The exchange of data between two SWCs over an unprotected communication link (which consists of *COM software* and *COM hardware* at the sender and the receiver, and *COM Link*) may be affected by faults, see Figure 3.1, "Unprotected communication". Such faults in the communication link can affect the integrity of exchanged data which may lead to a violation of the safety goal if such data is safety-related. Possible faults are:

- Random hardware faults, e.g. corrupt registers of a FlexRay controller.
- Transient faults, e.g. interference due to EMC on the physical link.
- Systematic faults within the communication software, e.g. an issue in the COM module.

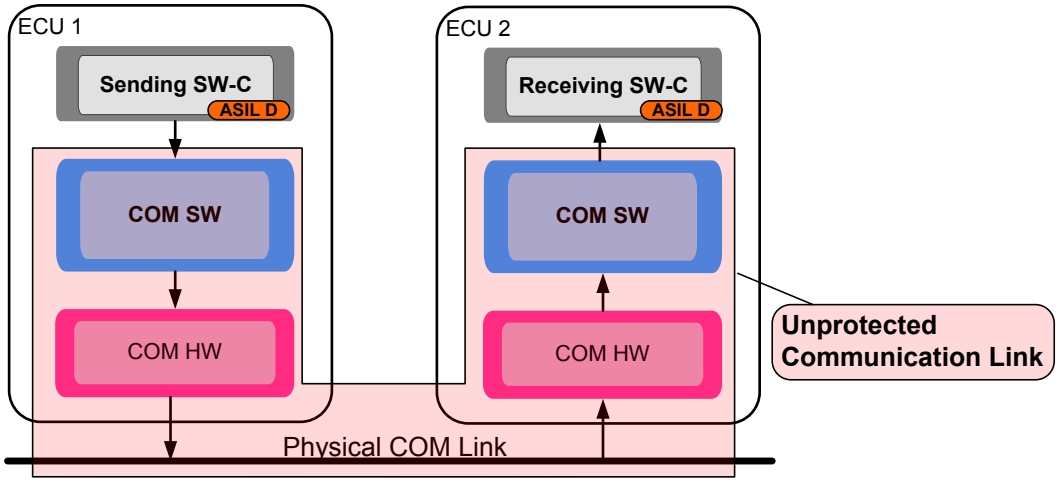

Figure 3.1. Unprotected communication

According to ISO 26262, part 6, clause D.2.4 [ISO26262_1ST] you can consider the causes for faults or effects of faults such as these listed below for each sender or each receiver with respect to the exchange of information, i.e. these faults can cause interference between software elements:

ID	Failure mode name
1	Repetition of information
2	Loss of information
3	Delay of information
4	Insertion of information
5	Masquerade or incorrect addressing of information
6	Incorrect sequence of information
7	Corruption of information
8	Asymmetric information sent from a sender to multiple receivers
9	Information from a sender received by only a subset of the receivers
10	Blocking access to a communication channel

Table 3.1. Causes for communication faults or effects of faults

3.1.2. Failure model of E2E communication

3.1.2.1. Term (object) definitions

This section defines all terms.

Term	Description	Synonym
Data source	A *data source*, i.e. SWC, is defined as an object that produces data.	Sender
Data sink	A *data sink*, i.e. SWC, is defined as an object that consumes data.	Receiver
Data element	A *data element* is defined as an object that is exchanged between sender and receiver.	
Communication link	A *communication link* is defined as a link between a data source and a data sink for the purpose of forwarding messages from the data source to the data sink.	Communication channel
Fault-free communication link	A *fault-free communication link* is defined as a communication link in the absence of faults.	
Message	A *message* is defined as data to be transmitted over a communication link.	Information
Safety-related message	A *safety-related message* is defined as a message that contains safety-related data.	

Term	Description	Synonym
Protected message	A *protected message* is defined as a message that contains user data and protection information, e.g. CRC, sequence counter, etc.	
Valid message	A *valid message* is defined as a protected message where the CRC fits to the user data of the message.	
Authenticated data source	An *authenticated data source* is defined as a data source that is uniquely identifiable by a data sink.	
Authentic message	An *authentic message* is defined as a message where the user data is identified as originated from the authenticated data source.	
Safety Transformer caller	A *caller* is defined as the module that calls the Safety Transformers.	Caller
Integrator	An *integrator* is defined as the person who integrates or configures the Safety Transformers.	
Communication error	A *communication error* is defined as occurrence of any E2E communication failure mode specified in Section 3.1.2.3, “E2E communication link failure modes”.	
Message sequence indicator	A *message sequence indicator* is defined as a number that represents the order of a message within a sequence of messages. For any two consecutively sent messages, `n` represents the first message and `n+1` the subsequent message.	
MaxSequenceCounter	The *MaxSequenceCounter* is defined as the maximum allowed value of the *Message sequence indicator*. Note: An infinite maximum number is not possible due to resource restrictions.	
DeltaCounter	The *DeltaCounter* is defined as the difference of the *Message sequence indicator* of a received valid and authentic message with respect to the last received valid and authentic message.	
Message loss window	A *message loss window* defines a *window size* where received valid and authentic messages are accepted if and only if the *DeltaCounter* is greater than zero and smaller than or equal to the *window size* plus one.	
Window size	The *window size* defines the size of the monitoring window on the receiver.	
MaxDeltaCounterInit	The *MaxDeltaCounterInit* is defined as the initial *window size* of the *message loss window*.	
Correct message	A *correct message* can be one of the following:	

Term	Description	Synonym
	▶ a valid message if authentication is not required depends on the context ▶ a valid and authentic message that is accepted by the *message loss window* if authentication is required.	
MaxNoNewOrRe-peatedData	The *MaxNoNewOrRepeatedData* is defined as the maximum allowed amount of continuously received messages with the failure modes *message loss* or *unintended message repetition* (see Section 3.1.2.3, “E2E communication link failure modes”) after which a deterministic detection of a correct message can be guaranteed.	
Max message loss window	The *max message loss window* is defined as the maximum allowed window size of the *message loss window* which equals *MaxDeltaCounterInit* + *MaxNoNewOrRepeatedData* - 1 with the constraint that (*MaxDeltaCounterInit* + *MaxNoNewOrRepeatedData* <= *MaxSequenceCounter*). Note: If the *message loss window* exceeds the *max message loss window*, then a deterministic detection of a correct message cannot be guaranteed anymore.	
SyncCounterInit	The *SyncCounterInit* is defined as the amount of data required to validate the consistency of the counter that must be received with a valid counter after an unexpected behavior is detected.	
Source-specific error indication	A *source-specific error indication* is defined as an indication to the caller of the cause of the communication error as defined in Section 3.1.2.3, “E2E communication link failure modes”.	
Communication status	A *communication status* is defined as communication-related information (states, operation modes, and errors) provided to the caller.	

Table 3.2. Term definition

3.1.2.2. Process definitions

This section defines all processes.

Term	Description	Synonyms
To transmit	*To transmit* is defined as the process to put a message into the communication link.	Send
To provide	*To provide* is defined as the process to make something available.	
To ignore	*To ignore* is defined as the process to tolerate discontinuities up to the specified maximum.	

Term	Description	Synonyms
To support	*To support* is defined as the process to contribute to provide something.	
To serialize	*To serialize* is defined as the process to transform a data element into a linear byte array.	
To deserialize	*To deserialize* is defined as the process to transform a linear byte array to a data element.	
To receive	*To receive* is defined as the process to get a message out of the communication link.	Deliver
To protect	*To protect* is defined as the process to add protection information, e.g. CRC, double inverse data to a message.	
to indicate	*To indicate* is defined as the process to return information, e.g. error information back to the caller of the Safety Transformers.	

Table 3.3. Process definition

3.1.2.3. E2E communication link failure modes

In the following, the failure modes with respect to an E2E communication link are defined with a mapping to the ISO failure mode ID of Table 3.1, “Causes for communication faults or effects of faults”.

Failure mode	ISO failure mode ID	Description
Unintended message repetition	1	*Unintended message repetition* is defined as receiving a message `m` more than once on the communication link on which `m` was transmitted once.
Message loss	2	*Message loss* is defined as never receiving message `m` on the communication link on which `m` was transmitted.
Insertion of messages	4	*Insertion of messages* is defined as receiving an unprotected message `m` on the communication link on which `m` was not transmitted. Note: Insertion of messages cannot be clearly identified at the receiver. If message m is interpreted as if it contains the protection information, i.e. CRC, a message corruption is detected.
Resequencing	6	*Resequencing* is defined as receiving message `m2` before message `m1` on the communication link on which message `m1` was transmitted before `m2`.

Failure mode	ISO failure mode ID	Description
Message corruption	7	*Message corruption* is defined as receiving `m'` instead of `m` and `m'` is not equal to `m` on the communication link on which `m` was transmitted.
Delayed reception	3	*Delayed reception* is defined as receiving a message `m` at `tR`, on which `tR` > `tE` > `tS`, `tE` is the latest expected reception time and `tS` is the time `m` that was transmitted.
Addressing fault	5	*Addressing fault* is defined as receiving a message `m` on a communication link B instead of the communication link A on which `m` was transmitted.
Masquerading	5	*Masquerading* is defined as receiving a non-authentic message `m` that appears authentic on the communication link on which `m` was transmitted or where `m` has been expected to be transmitted if the data source is identified by the communication link. Note: The latter case results from an addressing fault.

Table 3.4. Failure modes

NOTE

Failure mode IDs 8,9, and 10

For E2E communication the failure mode IDs 8 and 9 of Table 3.1, “Causes for communication faults or effects of faults” are not relevant as they are not related to a point-to-point communication scenario. Failure mode ID 10 of Table 3.1, “Causes for communication faults or effects of faults” (*blocking access to a communication channel*) results in the failure mode message delay, i.e. other messages cannot be received. Therefore, no additional detection mechanism is required.

3.1.3. Protected communication link

The detection of the failure modes at the receiving *SWC* requires the integration of safety mechanisms. These safety mechanisms include the transmission of related protection information to the safety-related user data by the sending *SWC* and the evaluation of this protection information data by the receiving SWC, see Figure 3.2, “Protected Communication”. Therefore AUTOSAR has specified a standard AUTOSAR library [ASRSWSE2E] that implements safety mechanisms for protected communication according to specific E2E Profiles.

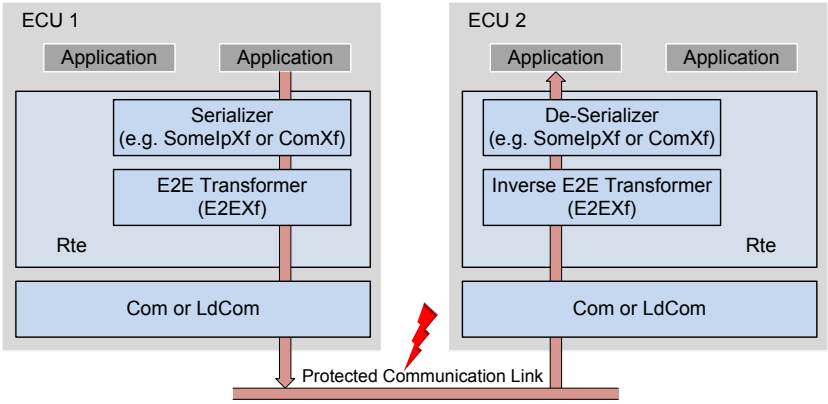

Figure 3.2. Protected Communication

A safety mechanism in the *E2EXf* detects communication faults. This includes functionality at the sender that usually adds protection information to the payload data, as well as functionality at the receiver side that processes the received information to detect a communication fault.

This *E2EXf* together with the *E2E* implements these safety mechanisms and is embedded by the *Safety Rte*. The *SomeIpXf* or *ComXf* modules perform the data serialization or deserialization.

3.1.4. Safety Transformers in AUTOSAR

The *Safety Transformers* product uses a new communication paradigm named *data transformation*, which was introduced in AUTOSAR version 4.2. Data transformation provides the means for efficient communication of large data elements of complex types and simplifies the configuration of additional data protection or security mechanisms.

In the sending path, a (complex) data element is first converted to a byte array by a so-called serializing transformer, e.g. *SOME/IP Transformer* or *ComXf* . This byte array is then provided to the *E2E Transformer* . This means that data is first serialized. Afterwards, the protection information e.g. CRC, sequence counter, Data Id, is calculated and appended to the serialized data stream. In the reception path, inverse functions are called in reverse order so that first the end-to-end protection information is verified and finally the byte array is transformed back into the (complex) data element. For the calculation of the end-to-end checksum, the *E2E Transformer* uses the protection routines of the *E2E*.

Figure 3.3, “Use cases 1 and 2” and Figure 3.4, “Use case 3” show the most common use cases for the *Safety Transformers*.

- Use case 1: *SOME/IP Transformer* and *LDCOM* for efficient data serialization to provide better support for large data as available with Ethernet.

- Use case 2: *SOME/IP Transformer* and *COM* to allow using the configuration variety of the *COM* module, e.g. filter mechanisms.
- Use case 3: *COM-based Transformer* and *COM* to allow using the serialization technology to ensure backward-compatibility on the network.

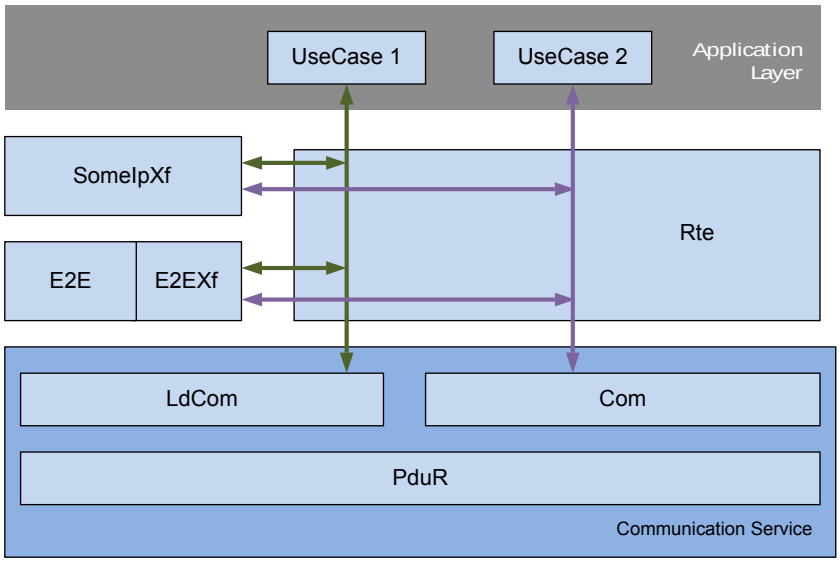

Figure 3.3. Use cases 1 and 2

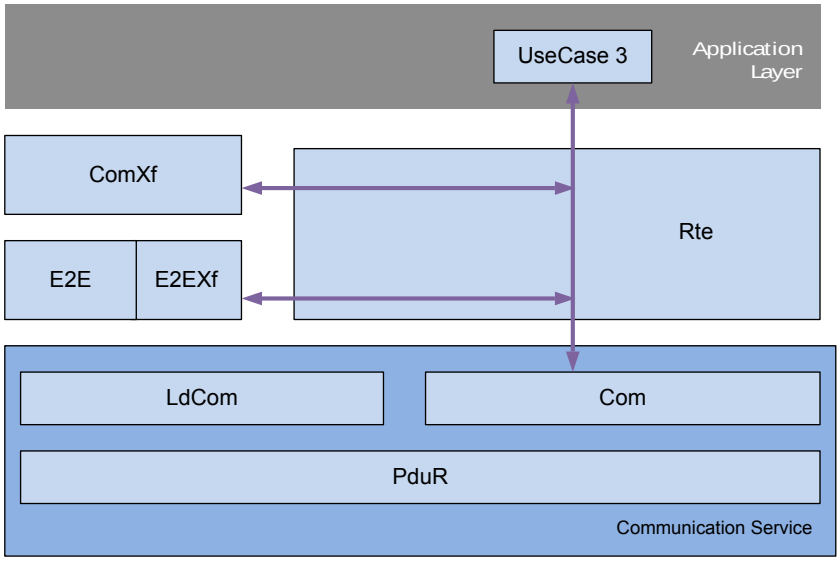

Figure 3.4. Use case 3

3.2. Description of Safety Transformers

3.2.1. Identification of the Safety Transformers

The Safety Transformers is composed of the following products:

- *EB tresos Safety E2E Transformer (E2E)*
- *EB tresos Safety E2E Transformer (SOME/IP)*
- *EB tresos Safety E2E Transformer (COM)*
- EB tresos Safety E2E Profiles, e.g. EB tresos Safety E2E Profile 5

Furthermore, the Safety Transformers contains a *Transformer Checker tool* and the following user documentation:

- EB tresos E2E Transformer (E2E) documentation [E2EXFUG]
- EB tresos AutoCore Generic 7 Transformer (SOME/IP) documentation [SOMEIPUG]
- EB tresos AutoCore Generic 7 Transformer (COM) documentation [COMXFUG]
- EB tresos AutoCore Generic 8 Transformers documentation [ACGTRANSFORMERUG]
- User documentation for the used E2E profiles
- Safety manual (this document)

3.2.2. Functional requirements of Safety Transformers

The Safety Transformers is developed as a safety element out of context (SEooC) according to INTERNATIONAL STANDARD ISO 26262 [ISO26262_1ST]. For more information, see Section 3.2.3, “Assumed safety requirements of Safety Transformers”.

3.2.3. Assumed safety requirements of Safety Transformers

The Safety Transformers is a safety element out of context (SEooC) on software level and implement the following assumed safety requirements. The correct implementation of these requirements has been verified.

3.2.3.1. Top level requirements for the Safety Transformers

3.2.3.1.1. EB_E2ESE000070: Freedom from interference of the data element exchange (S)

Id:	**EB_E2ESE000070**
Version:	1
Source:	EB
Status:	approved
Safety class:	ASIL-D
Short Description:	Freedom from interference of the data element exchange
Priority:	HIGH
Description:	The *Safety Transformers* shall provide freedom from interference of the data element exchange between two AUTOSAR SWCs.
Rationale:	E2E communication protection is state-of-the-art in safety-related automotive systems to ensure freedom from interference of the SWCs from the communication link in between. Detection of communication link failure modes is required to ensure freedom from interference.
Use Case:	E2E communication protection for communication on an unsafe communication link.
Dependencies:	None
Supporting Material:	
Verification criteria:	All communication link failure modes are detected by *Safety Transformers*, see Section 3.1.2.3, “E2E communication link failure modes”.
Comment:	

3.2.3.2. Claims for the Safety Transformers

3.2.3.2.1. EB_E2ESE000071: Protection of data element exchange (S)

Id:	**EB_E2ESE000071**
Version:	1
Source:	EB
Status:	approved
Safety class:	ASIL-D
Short Description:	Protection of data element exchange
Priority:	HIGH
Description:	The *Safety Transformers* shall provide protection of data element exchange between two AUTOSAR SWCs.
Rationale:	To detect communication link failures, the data element must be protected.
Use Case:	E2E communication protection for communication on an unsafe communication link.
Dependencies:	EB_E2ESE000072
Supporting Material:	
Verification criteria:	The data element exchange is protected by adding appropriate protection information.
Comment:	

3.2.3.2.2. EB_E2ESE000072: Communication failure detection (S)

Id:	**EB_E2ESE000072**
Version:	1
Source:	EB
Status:	approved
Safety class:	ASIL-D
Short Description:	Communication failure detection
Priority:	HIGH
Description:	The *Safety Transformers* shall provide communication failure detection of protected data element exchange between two AUTOSAR SWCs.
Rationale:	Detection of communication link failures.
Use Case:	E2E communication protection for communication on an unsafe communication link.
Dependencies:	EB_E2ESE000071
Supporting Material:	
Verification criteria:	All relevant communication link failure modes are detected by the *Safety Transformers*.
Comment:	

3.2.3.2.3. EB_E2ESE000073: Configurable tolerance level (S)

Id:	**EB_E2ESE000073**
Version:	1
Source:	EB
Status:	approved
Safety class:	ASIL-D
Short Description:	Configurable tolerance level
Priority:	HIGH
Description:	The *Safety Transformers* shall provide a configurable tolerance level regarding communication faults.
Rationale:	Appropriate robustness against limited message loss between two SWCs.
Use Case:	Communication between applications which tolerate lost messages.
Dependencies:	None
Supporting Material:	
Verification criteria:	Communication is not interrupted due to message sequence discontinuities within the tolerance level.
Comment:	

3.2.3.2.4. EB_E2ESE000074: Source-specific error indication (S)

Id:	**EB_E2ESE000074**
Version:	1
Source:	EB
Status:	approved
Safety class:	ASIL-D
Short Description:	Source-specific error indication
Priority:	HIGH
Description:	The *Safety Transformers* shall provide source-specific error indication and communication status to the Safety Transformer caller.
Rationale:	Detected errors cannot be handled by the *Safety Transformers* and thus must be indicated to the Safety Transformer caller.
Use Case:	The application performs individual error handling on the error indication and communication status that is indicated by the *Safety Transformers*.
Dependencies:	None
Supporting Material:	
Verification criteria:	Source-specific error indication and communication status are indicated to the Safety Transformer caller via a function return value.
Comment:	

3.2.3.2.5. EB_E2ESE000100: Efficient communication of large data elements (S)

Id:	**EB_E2ESE000100**
Version:	1
Source:	EB
Status:	approved
Safety class:	ASIL-D
Short Description:	Efficient communication of large data elements
Priority:	HIGH
Description:	The *Safety Transformers* shall be able to serialize data elements into a network representation (and vice versa) independent of different kinds of operating systems and hardware and optimized for serialization speed.
Rationale:	A defined representation of data on the bus allows the interoperability of transformers of different vendors. An optimization for serialization speed induces a enhanced bandwidth which is needed for Ethernet communication.
Use Case:	Efficient data element transformation for fast networks and relocation of SWCs to other ECUs is simplified.
Dependencies:	None
Supporting Material:	
Verification criteria:	A data element has been transformed to support data rates used in Ethernet networks.
Comment:	

3.2.4. Safety mechanism used by Safety Transformers

The safety mechanism of the transformer enables the receiver to detect communication faults. This safety mechanism consists of protection information that contains a data checksum, a sequence identifier, and a message identifier. The sender adds protection information to the data for transmission and the receiver evaluates the received protection information to detect and indicate communication faults. Within the transformer concept of AUTOSAR, a transformer gets a data buffer that might already consist of some payload with header information. The transformer computes protection information, and adds this information within an E2E header to the buffer Figure 3.5, “E2E Header”.

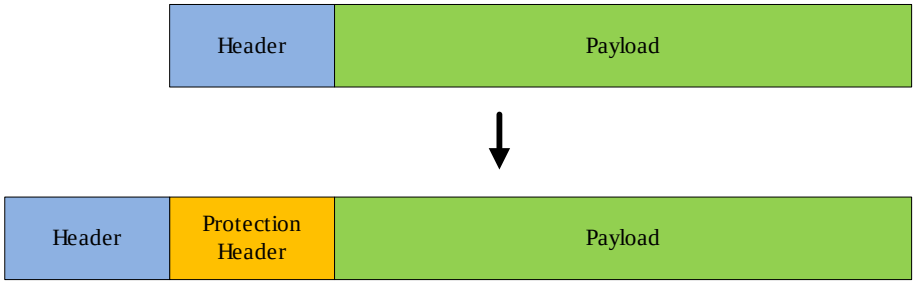

Figure 3.5. E2E Header

3.2.5. System context of the EB tresos Safety E2E Transformers

The system context shows the interaction of the *EB tresos Safety E2E Transformers* with other components.

The system context of the *EB tresos Safety E2E Transformers* is depicted in Figure 3.6, "EB tresos Safety E2E Transformers system context".

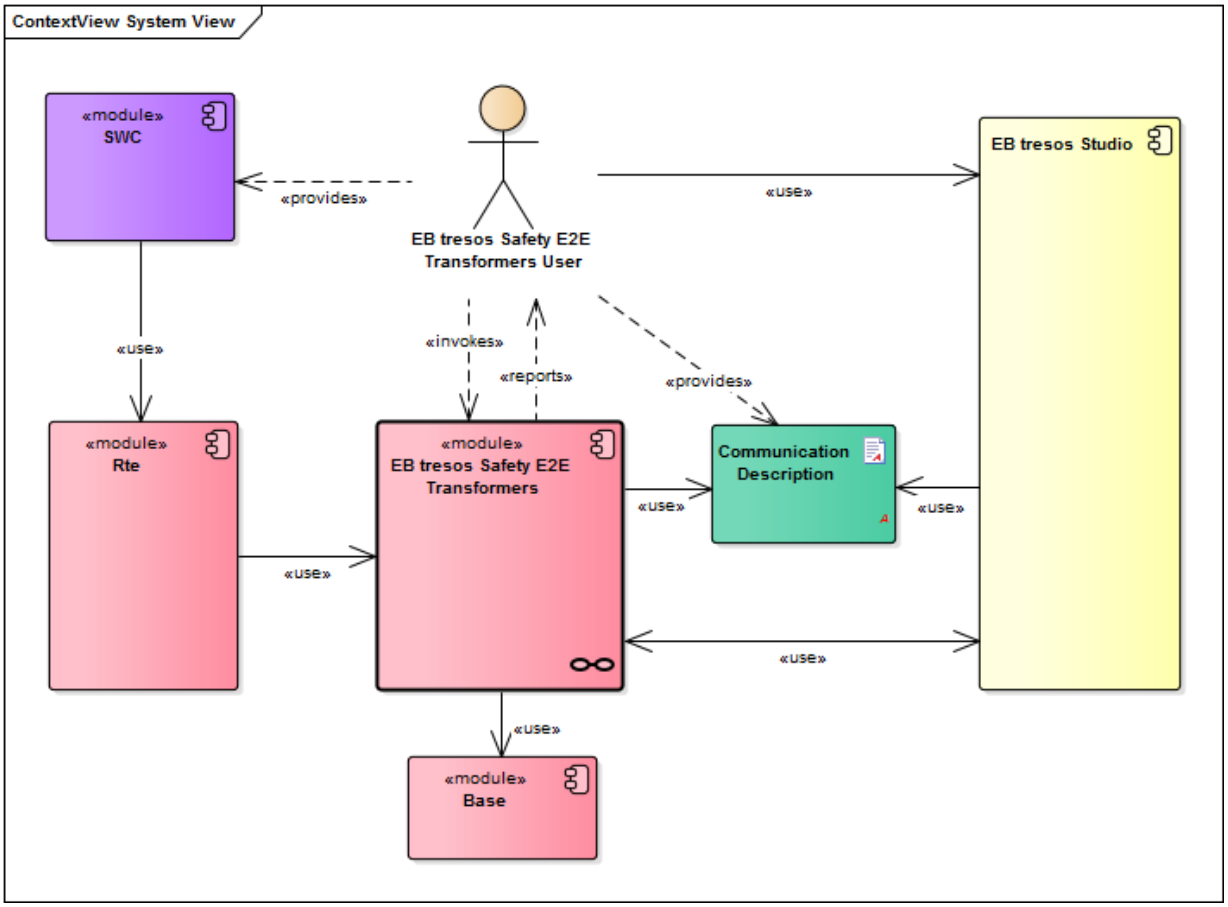

Figure 3.6. EB tresos Safety E2E Transformers system context

The *EB tresos Safety E2E Transformers* interacts with the following components:

Component	Description
Rte	The Run-Time Environment (RTE) uses the transformation services of the EB tresos Safety E2E Transformers to linearize and protect communication between AUTOSAR software-components.
SWC	The EB tresos Safety E2E Transformers User provides an AUTOSAR application software providing the data which are subject to transformation by the EB tresos Safety E2E Transformers.
EB tresos Studio	The EB tresos Studio is responsible for reading the system configuration and extracting the ECU specific settings. The EB tresos Studio also provides an environment for configuring the EB tresos Safety E2E Transformers and triggers the code generators.
Communicaton Description	The EB tresos Safety E2E Transformers User provides the system configuration.
EB tresos Safety E2E Transformers User	The EB tresos Safety E2E Transformers User integrates the EB tresos Safety E2E Transformers in the system.
Base	The Base component provides standardized header files to define common types.

3.2.6. Structure of the Safety Transformers BSW

The *Safety Transformers* is internally structured as depicted in (Figure 3.7, “Composition view”).

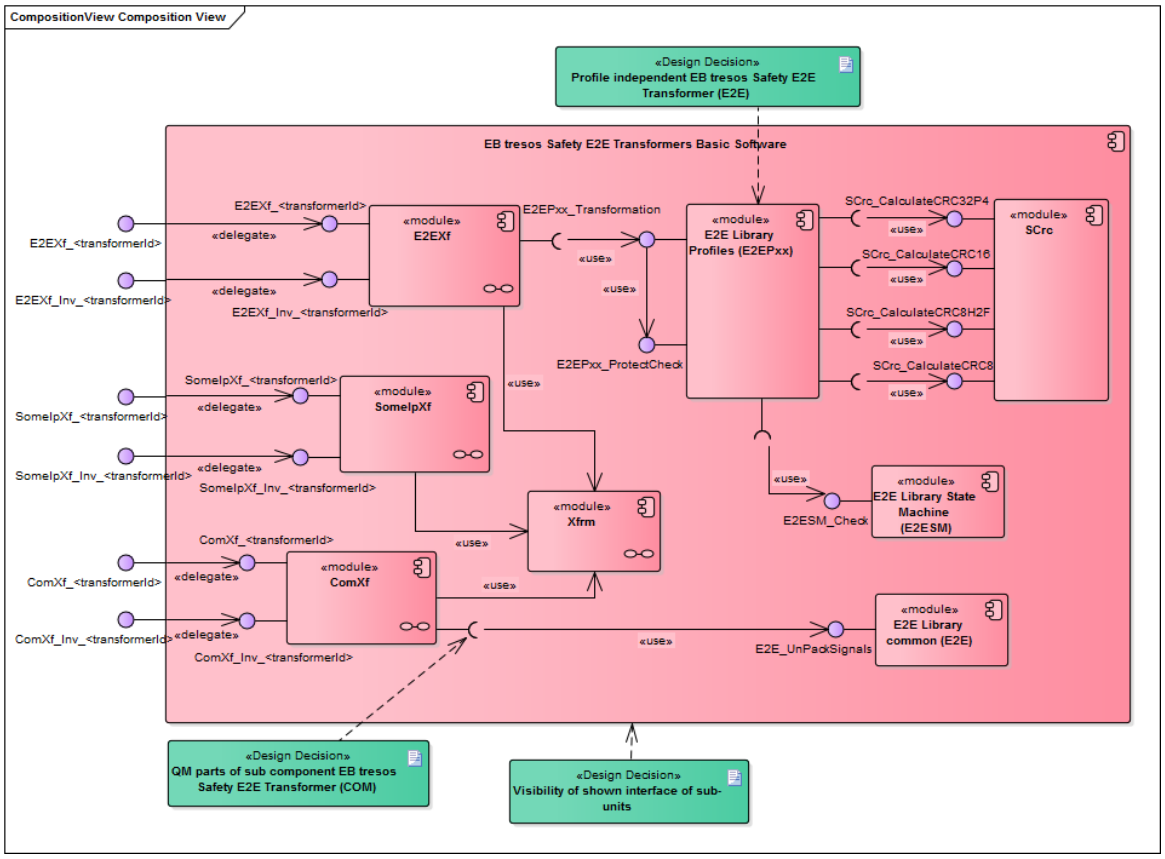

Figure 3.7. Composition view

Component	Description
E2E Library common (E2E)	The *E2E Library common (E2E)* provides an implementation of common used constructs for the profile dependent part *E2E Library Profiles (E2EPxx)* as well as a set of macros to serialize and deserialize data bit aligned.
E2E Library Profiles (E2EPxx)	The *E2E Library Profiles (E2EPxx)* provides an implementation of customizable protection and verification mechanism for the End-to-End transmission of signals as well as the profile dependent part used by *EB tresos Safety E2E Transformer (E2E).*
E2E Library State Machine (E2ESM)	The *E2E Library State Machine (E2ESM)* provides the implementation of a customizable state machine forming a resulting reception state out of several consecutive verification results determined by *E2E Library Profiles (E2EPxx).*
ComXf	The *EB tresos Safety E2E Transformer (COM)* provides wrapper functions which use the post build configuration and libraries of the Com module. The component *EB tresos Safety E2E Transformer (COM)* follows the functional description in the AUTOSAR document *COM Based Transformer*. The composition is described in a dedicated detailed design document.

E2EXf	The *EB tresos Safety E2E Transformer (E2E)* provides functionality for protecting safety-related data elements in a transmission. It provides interfaces for data protection at sender side and for data check at receiver side. The component *EB tresos Safety E2E Transformer (E2E)* follows the functional description in the AUTOSAR document *E2E Transformer*. The composition is described in a dedicated detailed design document.
SCrc	The *Secure CRC* provides a collection of safe CRC calculation routines.
SomeIpXf	The *EB tresos Safety E2E Transformer (SOME/IP)* provides functionality to serialize and deserialize data with the SOME/IP on-the-wire format and specifies a mechanism for Client/Server communication. The component *EB tresos Safety E2E Transformer (SOME/IP)* follows the functional description in the AUTOSAR document *SOME/IP Transformer*. The composition is described in a dedicated detailed design document.
Xfrm	The *Xfrm* module provides mechanism shared among all transformers.

3.2.7. Dynamic behavior of the Safety Transformers

3.2.7.1. Interaction view

Note: `INOUT` and `OUT` parameters of a message in the sequence chart diagrams are depicted as pointer to pointer (`**`).

3.2.7.1.1. Initialization of Safety Transformers

Figure 3.8, “Safety Transformers Initialization” shows the initialization of the Safety Transformers.

[EB_E2ESE.safetyanalysis.Initialization.Sequence]

Before using data protection or verification the software component or integration code must call the initialization sequence as shown in the following sequence diagram in arbitrary order.

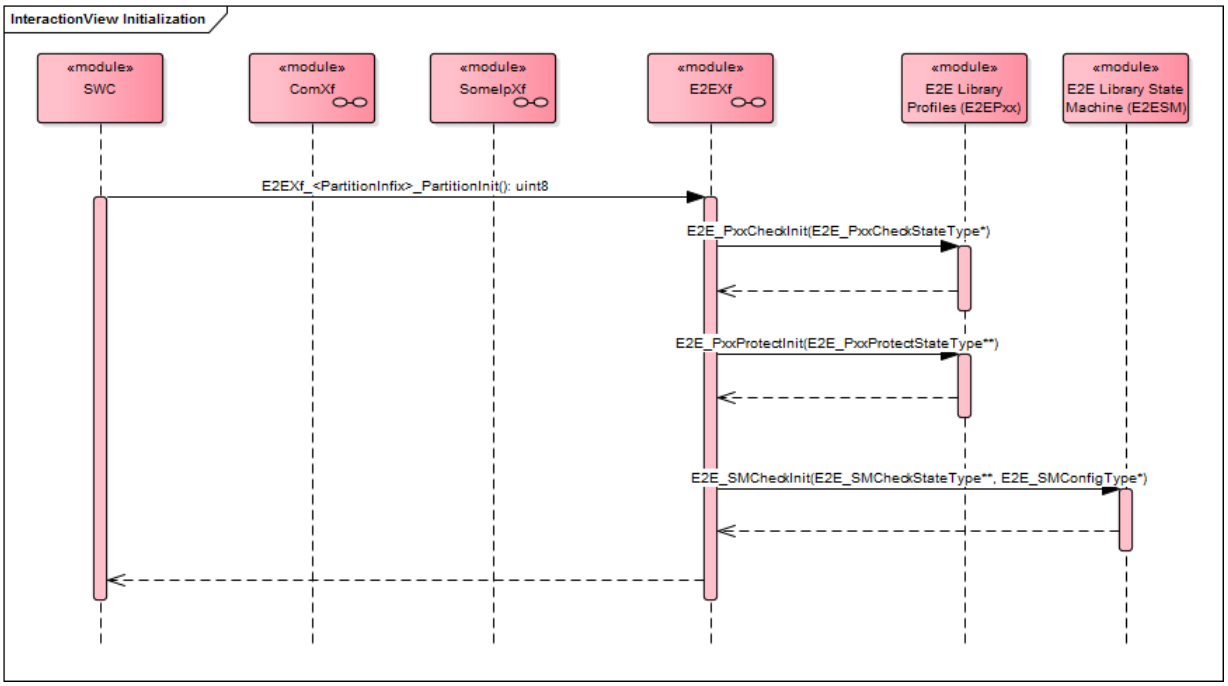

Figure 3.8. Safety Transformers Initialization

3.2.7.1.2. Use case: ComXf

This section shows the interaction of components for the use case COM-based Transformer. This maps to use case 3 as descripted in Section 3.1.4, “Safety Transformers in AUTOSAR”.

3.2.7.1.2.1. ComXf seen in context

Requirements: EB_E2ESE.SwAD.030 , EB_E2ESE.SwAD.031

Figure 3.9, “Context interaction for transmission of protected data with use case COM-based Transformer” shows protection for data transmission using the COM-based Transformer in the context of *EB tresos Safety E2E Transformers*

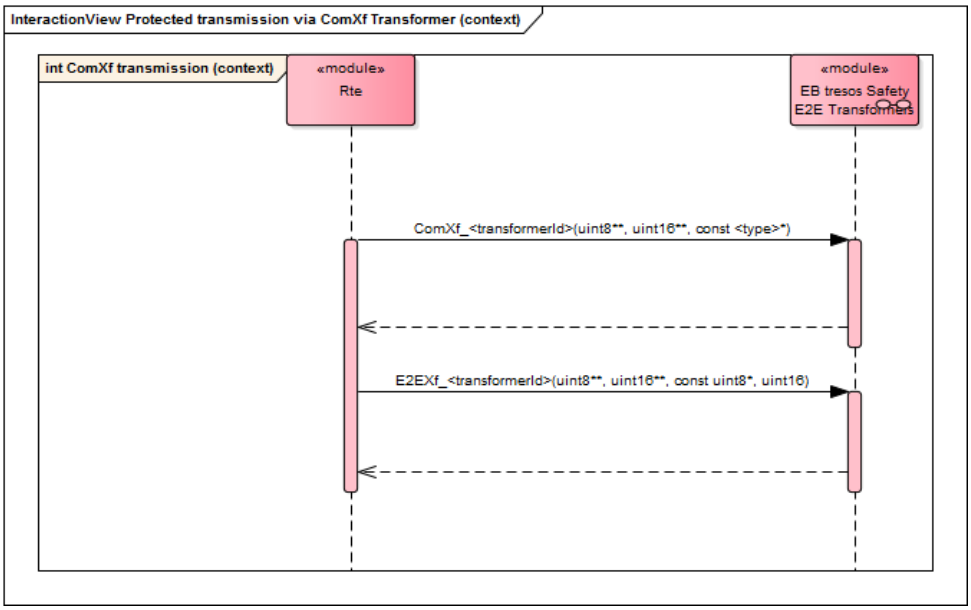

Figure 3.9. Context interaction for transmission of protected data with use case COM-based Transformer

Figure 3.10, “Context interaction for reception of protected data with use case COM-based Transformer” shows the verification of data reception using the COM-based Transformer in the context of *EB tresos Safety E2E Transformers*

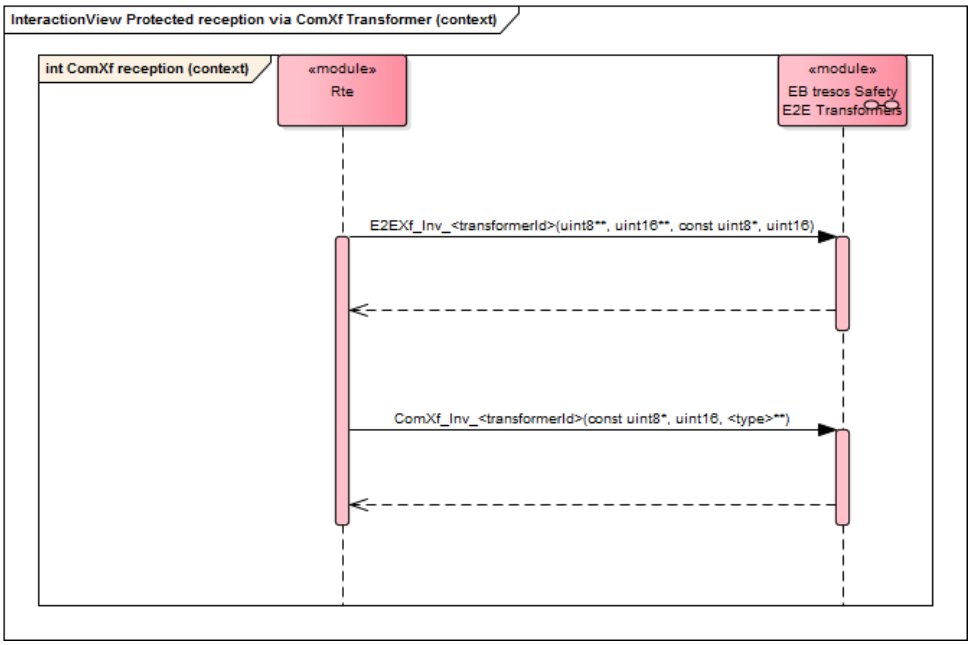

Figure 3.10. Context interaction for reception of protected data with use case COM-based Transformer

3.2.7.1.2.2. ComXf seen in composition

Requirements: EB_E2ESE.SwAD.004 , EB_E2ESE.SwAD.006

Figure 3.11, “Composition interaction for transmission of protected data with use case COM-based Transformer” shows protection for data transmission using the COM-based Transformer showing the composition of *EB tresos Safety E2E Transformers*. This example view shows the interface used for in-place *transformation*.

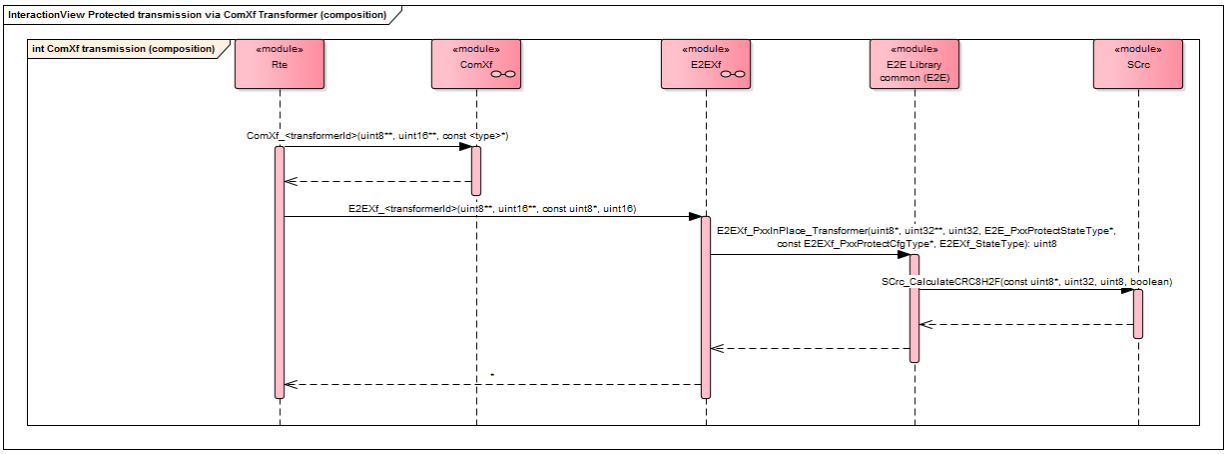

Figure 3.11. Composition interaction for transmission of protected data with use case COM-based Transformer

Figure 3.12, “Composition interaction for reception of protected data with use case COM-based Transformer” shows the verification of data reception using the COM-based Transformer showing the composition of *EB tresos Safety E2E Transformers*. This example view shows the interface used for in-place *transformation*.

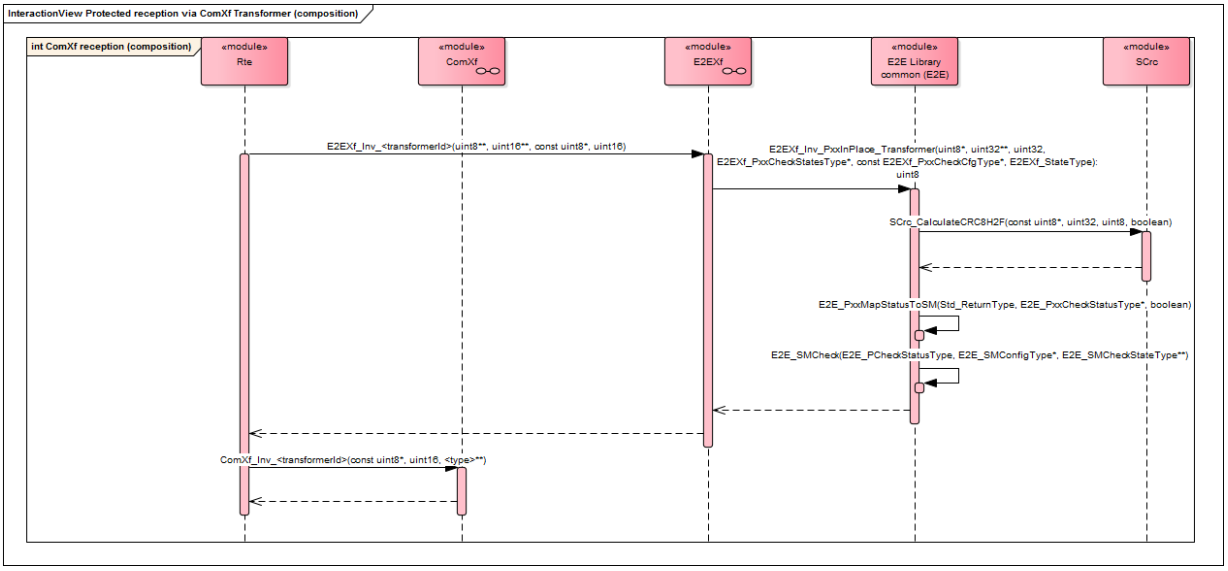

Figure 3.12. Composition interaction for reception of protected data with use case COM-based Transformer

3.2.7.1.3. Use case: SomeIpXf

This section shows the interaction of components for the use case SOME/IP Transformer. This maps to use case 1 and 2 as descripted in Section 3.1.4, “Safety Transformers in AUTOSAR”.

3.2.7.1.3.1. SomeIpXf seen in context

Requirements: EB_E2ESE.SwAD.029 , EB_E2ESE.SwAD.031

Figure 3.13, “Context interaction for transmission of protected data with use case SOME/IP Transformer” shows the protection of data transmission using the SOME/IP Transformer in the context of *EB tresos Safety E2E Transformers*.

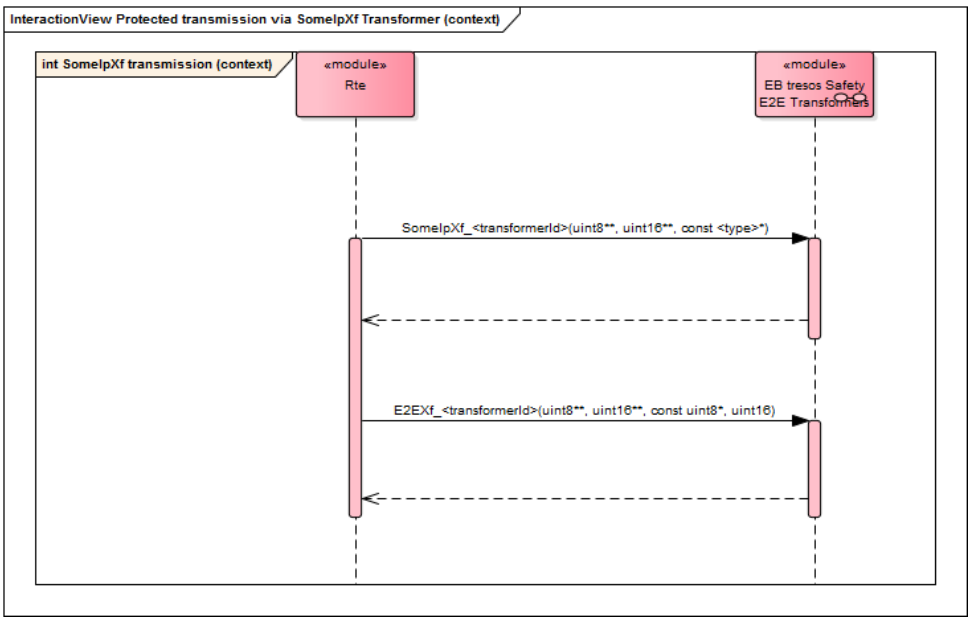

Figure 3.13. Context interaction for transmission of protected data with use case SOME/IP Transformer

Figure 3.14, “Context interaction for reception of protected data with use case SOME/IP Transformer” shows the verification of data transmission using the SOME/IP Transformer in the context of *EB tresos Safety E2E Transformers*.

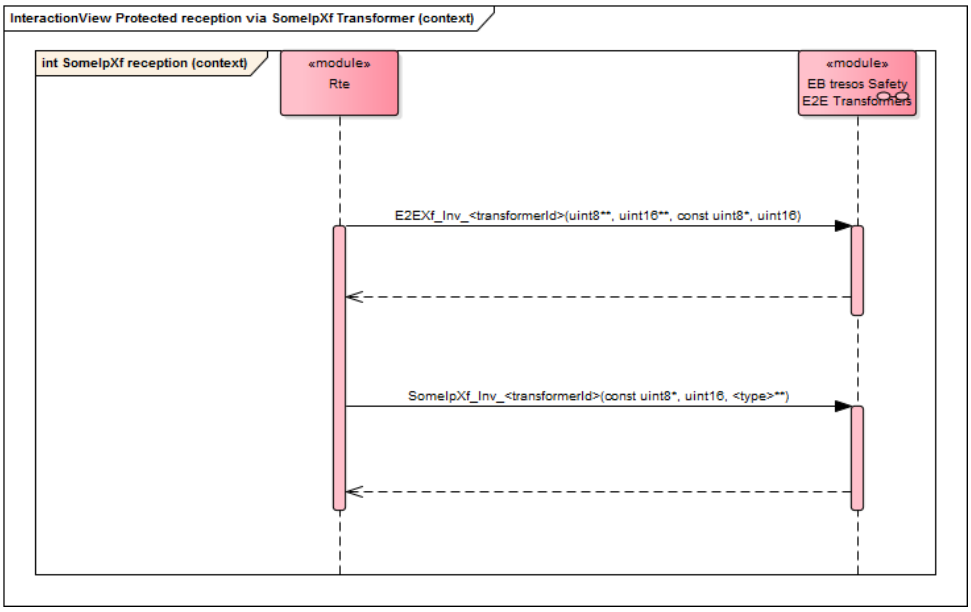

Figure 3.14. Context interaction for reception of protected data with use case SOME/IP Transformer

3.2.7.1.3.2. SomeIpXf seen in composition

Requirements: EB_E2ESE.SwAD.002 , EB_E2ESE.SwAD.006

Figure 3.15, “Composition interaction for transmission of protected data with use case SOME/IP Transformer” shows the protection of data transmission using the SOME/IP Transformer showing the composition of *EB tresos Safety E2E Transformers*. This example view shows the interface that is used for out-of-place *transformation*.

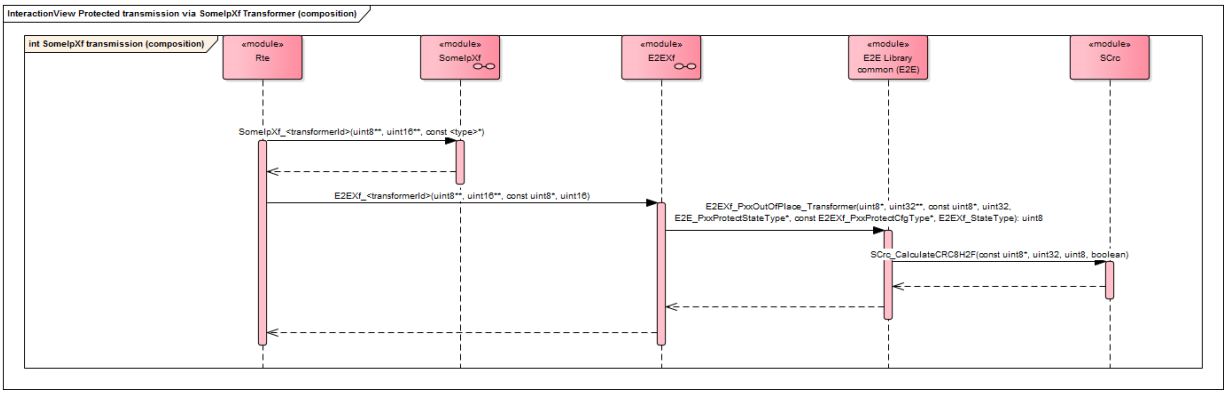

Figure 3.15. Composition interaction for transmission of protected data with use case SOME/IP Transformer

Figure 3.16, “Composition interaction for reception of protected data with use case SOME/IP Transformer” shows the verification of data reception using the SOME/IP Transformer showing the composition of *EB tresos Safety E2E Transformers*. This example view shows the interface that is used for out-of-place *transformation*.

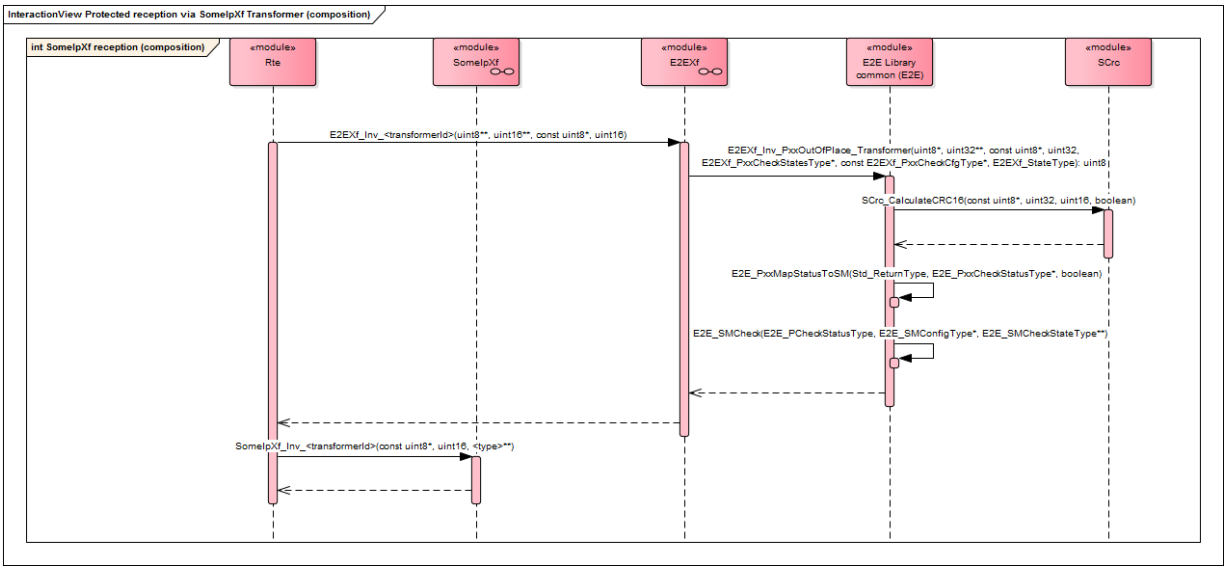

Figure 3.16. Composition interaction for reception of protected data with use case SOME/IP Transformer

3.2.8. Robustness of Safety Transformers

3.2.8.1. Robustness against hardware faults

Safety Transformers is not robust against hardware faults. It is assumed that the hardware uses fault detection mechanisms such as dual redundant processing, e.g. lock-step mode, error detection, and correction codes (ECC) etc.

3.2.8.2. Robustness against systematic software errors

The Safety Transformers is verified in accordance to [ISO26262_1ST] to detect systematic software errors in the Safety Transformers code.

There are no means implemented in Safety Transformers which make them robust against systematic errors in the calling software.

3.2.8.3. Robustness against configuration errors

Safety Transformers is susceptible to configuration errors. Configuration errors of the Safety Transformers must be handled on system level. The integrator must ensure to configure the Safety Transformers correctly. For more details refer to Chapter 6, “Configuration verification criteria“.

3.2.8.4. Robustness against resource conflicts

Safety Transformers is not robust against resource conflicts. The integrator must ensure that sufficient resources are available. For more details refer to e.g. EB_E2ESE065132.

3.2.8.5. Robustness against interrupt overload

Safety Transformers is not robust against interrupt overload. The integrator must ensure to configure the system to avoid interrupt overload or it is handled without violating the safety relevant parts.

3.2.8.6. Robustness against input errors

Safety Transformers must be used in accordance to the Safety Transformers safety manual. Input errors caused by incorrect use of Safety Transformers API functions must be avoided by verifying the caller according to its safety integrity level.

3.2.8.7. Robustness against data starvation

Safety Transformers is not robust against data starvation. The integrator must consider the case that no new data are provided to the Safety Transformers or received by the Safety Transformers on system level.

3.2.9. What Safety Transformers does not does

Safety Transformers does not validate the provided *Communication Description*. Safety Transformers cannot prevent communication errors, but only detect them.

3.2.10. Backward compatibility

The backward compatibility of the Safety Transformers is documented in the release notes.

3.2.11. Change control

Problems and change requests can be reported to the distributor. For more information, contact the technical support.

3.2.12. Limitations of Safety Transformers

The limitations are defined in the module user guides or in the release notes.

4. Using the Safety Transformers

Safety Transformers is developed as a safety element out of context (SEooC). Therefore, Elektrobit Automotive GmbH assumes that the environment meets particular requirements so that the Safety Transformers code behaves appropriately and safely.

Each release of *Safety Transformers* is developed with and tested against a particular AutoCore version, which is specified within the Safety Transformers, see [RELNOTES].

For more information on intended usage and possible misuse of *Safety Transformers*, see the module user guides.

4.1. Prerequisites

The *Safety Transformers* requires the following components:

- *EB tresos Studio*
- *System Description*

The product version of EB tresos Studio that is supported by this release is documented in the following user's guide:

- EB tresos E2E Transformer (E2E) documentation [E2EXFUG]
- EB tresos AutoCore Generic 7 Transformer (SOME/IP) documentation [SOMEIPUG]
- EB tresos AutoCore Generic 7 Transformer (COM) documentation [COMXFUG]
- EB tresos AutoCore Generic 8 Transformers documentation [ACGTRANSFORMERUG]

4.2. Installing Safety Transformers

Before you install *Safety Transformers*:

- Make sure that the correct EB tresos Studio version is already installed.

To install *Safety Transformers* follow the steps described in chapter *Installing additional products to EB tresos Studio* in the EB tresos installation guide which is part of the file which you downloaded from EB Command.

NOTE **Install all packages**

You must install all packages from the file which you have downloaded.

After installation with `setup.exe` from the `$TRESOS_BASE/` the following folders are created:

- <TRESOS_BASE\doc\7.0_EB_tresos_Safety_E2E_Protection

  Contains the documentation files for the Safety Transformers

  - Safety_E2E_Transformers_release_notes.pdf
  - Safety_E2E_Transformers_safety_manual.pdf
- <TRESOS_BASE>\plugins

  Contains all installed module plugins in this folder of the Safety Transformers
- <TRESOS_BASE>\tools

  Contains all installed module tools in this folder of the Safety Transformers
- <TRESOS_BASE>\checksums

  Contains all SHA-1 hash files of a module

4.3. Verifying the integrity of the Safety Transformers sources

The Safety Transformers delivery contains the checksums of all files that are part of the shipped package. You can find these files in `$TRESOS_BASE/checksums` folder of each module.

Verify for each module that the SHA-1 hash of the file `<module><version>.sha1.sha1` (i.e. `E2EXf_TS_TxDxM1I0R0.sha1.sha1`) matches to the corresponding SHA-1 hash of the reference file <version>README.txt. The <version>README.txt file is delivered via EB command.

If the delivery is not directly received from Elektrobit Automotive GmbH, but from an authorized distributor (e.g. an OEM with the right to issue sublicences), then the customer and/or distributor must assure that the delivery process is not affected (no loss of information).

WARNING **Use Safety Transformers only with correct files**

Use Safety Transformers only if all files are verified and intact.

In case of corrupted files, reinstall the package that contains Safety Transformers, and watch out for any warnings that appear during this process. Contact Elektrobit Automotive GmbH support if the verification still fails.

4.4. Field monitoring

Periodically, a list of all related problem reports, i.e. EB tresos AutoCore known issues is automatically created for each delivery that is under maintenance and provided via EB Command. The list provides the following information:

- Related release version.
- A list of published issues with the following information:
  - Unique issue ID
  - Affected module and version
  - Defect description
  - Workaround if applicable

The access to the known issues list is protected by a customer-specific login and password, the EB Command login.

4.5. Correct use and integration of the Safety Transformers

This section defines the correct use and integration which are necessary to use and integrate the Safety Transformers in a safe way.

4.5.1. Tooling workflow of the Safety Transformers

The use of Safety Transformers is split into two steps

- Generating of the Safety Transformers code.
- Verifying of the generated Safety Transformers code.

Figure 4.1, “EB tresos Safety E2E Transformers workflow” shows the tooling workflow and the different activities allocated to the participating systems for the configuration and generation of the *EB tresos Safety E2E Transformers*.

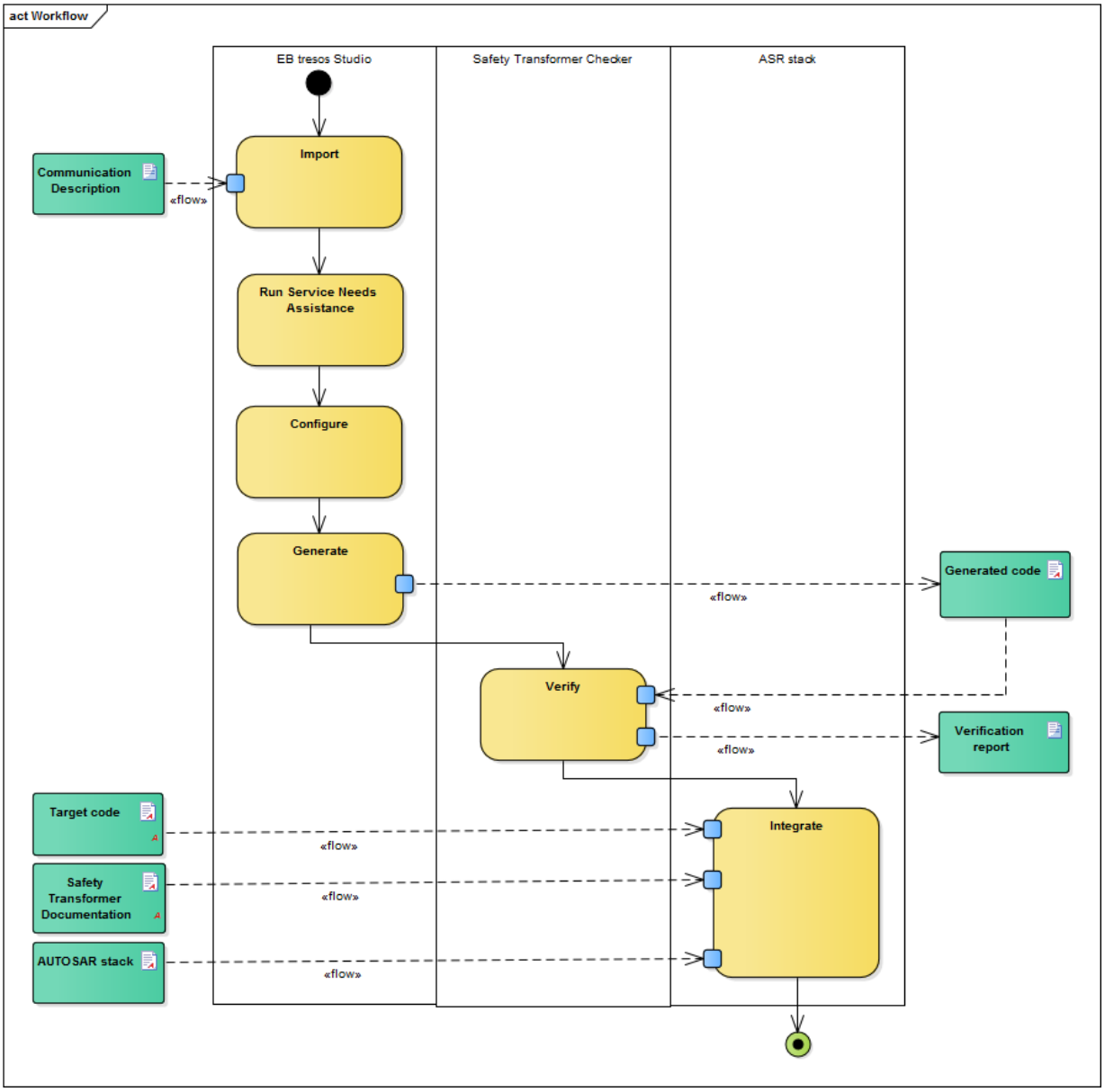

Figure 4.1. EB tresos Safety E2E Transformers workflow

Element	Description
Input *Communication Description*	The *Communication Description* is an ECU extract that conforms to the specification of the AUTOSAR system and SWC template. Note that splittable elements can be distributed on multiple physical files. The *Communication Description* consists of the following content: ► One instance of the entity `SystemTemplate::System`

	▶ Instances of the entity `CoreTopology::EcuInstance` ▶ Mapping of SWC prototypes to specifice ECUs using the SwcToEcuMapping class ▶ Mapping of `VariableDataPrototype` with complex data types to `SystemSignals(SenderReceiverToSignalGroupMapping)` ▶ E2E Protection information
Workflow step *Import*	A new *EB tresos Studio* project for an *ECU* is created that contains at least a serialization module, i.e. SOME/IP Transformer or COM-based Transformer, E2E Transformer SCrc, E2E Library, E2E Profiles XX (XX stands for the delivered profiles, e.g. 02). The *Communication Description* is imported via the Importer/Exporter dialog.
Workflow step *Run Service Needs Assistant*	The wizard `Service Needs Assistant` of *EB tresos Studio* automatically push required configurations into the SOME/IP Transformer, COM-based Transformer or E2E Transformer.
Workflow step *Configure*	Specific configuration paramter can be adapted in the module configuration window.
Workflow step *Generate*	The Safety Transformers Generator generate header files and source files that contain transformation functions to serialize or deserialize and protect data elements.
Input/output *Generated code*	The generated code and the configuration data are the configuration dependent part of the Safety Transformers Basic Software and running on the target platform.
Workflow step *Verify*	The generated code and the configuration data are verified with the Safety Transformers Checker, by validating them against the generated code.
Output *Verification report*	The *Verification report* is the output of the Safety Transformers Checker and contains the following information for the user: ▶ The result of the verification of the *Generated code* ▶ Execution status of the Safety Transformers Checker ▶ A number of review rules ▶ A detailed execution log of the Safety Transformers Checker
Input *Target code*	The generated code, the configuration data and the static code form the Safety Transformers Basic Software of the *EB tresos Safety E2E Transformers* on the target platform.
Input *Safety Transformer Documentation*	The *EB tresos Safety E2E Transformers* documentation contains information about how to integrate the target code into the AUTOSAR stack.

Input *AUTOSAR stack*	The AUTOSAR stack is the missing part of the target. The AUTOSAR stack and the Safety Transformers Basic Software form the software that runs on the target platform.
Workflow step *Integrate*	The *EB tresos Safety E2E Transformers* can be integrated into the AUTOSAR communication stack.

Table 4.1. Tooling workflow description

5. Safety element out of context (SEooC) definition

The Safety Transformers product is developed as a Safety Element out of Context (SEooC) according to [ISO26262-10_DIS].

5.1. Functional scope of the SEooC

5.1.1. Provided functionality

The functional scope of the SEooC is defined by the requirements as specified in Section 3.2.3, “Assumed safety requirements of Safety Transformers”.

WARNING	**Assumed safety requirements** The requirements listed in Section 3.2.3, “Assumed safety requirements of Safety Transformers” are not sufficient to fulfill all safety requirements imposed on the software architecture of the entire system. Depending on the system application, additional safety mechanisms may be necessary to guarantee system safety.

5.1.2. Assumed employment

The Safety Transformers product assumes that you employ all of its parts according to the specification that is provided in the safety manual.

5.1.3. Assumed external functionality

The Safety Transformers assumes that the following safety mechanisms are used on the ECU to ensure the integrity of safety-related functionality:

- Safe API calls of the Safety Transformers, e.g. provided by *EB tresos Safety Rte*
- Memory protection for spatial freedom from interference, e.g. provided by EB tresos Safety OS
- Control flow monitoring, e.g. provided by *EB tresos Safety TimE*

6. Configuration verification criteria

6.1. Configuration Rules

6.1.1. Workflow step: Configuring

In the following, the rules for the workflow step *Configure* are specified:

[EB_E2ESE062001]

Verify that all parameters listed below that are included in the `XfrmImplementationMappings`, container are configured.

- XfrmTransformerBswModuleEntryRef
- XfrmInvTransformerBswModuleEntryRef
- XfrmTransformationTechnologyRef
- XfrmIsSafetyTransformer
- XfrmOsApplicationRef.

For more details refer to [E2EXFUG].

[EB_E2ESE062005]

Verify that the configuration parameter `XfrmIsSafetyTransformer` is set to `DISABLED`).
Note: If the `XfrmIsSafetyTransformer` parameter is set to `DISABLED` and the E2E transformer is in the transformer chain, all transformer code is safe within this partition.

[EB_E2ESE062003]

Verify that the receiver-specific E2E Description parameters, i.e maxDeltaCounterInit, maxNoNewOrRepeatedData, and syncCounterInit, are equal for all protected receiver ports that are associated with the same E2E Protection and the same SWC. Rationale: parameters are defined as macros that use the short name of the E2E Description. Therefore multiple definitions are not possible.

[EB_E2ESE062007]

Verify for the SomeIp transformer configuration that the size of length fields of fixed size arrays is 1, 2 or 4 bytes.

[EB_E2ESE062008]

Verify for the SomeIp transformer configuration that the size of length fields of structures is 1, 2 or 4 bytes.

[EB_E2ESE062009]

Verify for the SomeIp transformer configuration that the length of size indicators of variable size arrays of type linear, square or rectangular is 1, 2, 4 or 8 bytes.

[EB_E2ESE062010]

Verify for the SomeIp transformer configuration that the length of size indicators of variable size arrays of type fully flexible is 1, 2 or 4 bytes.

[EB_E2ESE062011]

Verify for the SomeIp transformer configuration that the maximum nesting level of structures in fixed size arrays is 10.

[EB_E2ESE062012]

Verify for the SomeIp transformer configuration that the maximum number of dimensions of variable size arrays of type fully flexible is 3.

[EB_E2ESE062013]

Verify for the SomeIp transformer configuration that the pre-compile time parameter "XfrmBuffer-LengthType" is configured to "UINT16".

6.1.2. Workflow step: Generating

In the following, the rules for the workflow step *Generate* are specified:

[EB_E2ESE062004]

Verify that all warning and error messages of the code generation are analyzed.

6.1.3. Workflow step: Verifying generated files

In the following, the rules for the workflow step *Verify generated files* are specified:

[EB_E2ESE062006]

Verify for the used safety related transformers the generated code with the check tool. Therefore, generate the Safety Check Report for each used safety related transformer. Verify that the result in the Safety Check Report indicates "PASSED". Perform the list of reviews documented in the Safety Check Report. For usage instructions of the check tool see Appendix B, “Safety Checker“.

6.2. Assumptions of Safety Transformers

This following section defines assumptions on the item under which Safety Transformers can be used.

The *EB tresos Safety E2E Transformer* has no specific safety requirements on the item compared to common SWCs, i.e. there are no additional safety requirements on the item.

NOTE **Freedom from interference**

In a typical usage scenario, the Safety Transformers is integrated on one microcontroller together with non-safety-related components. In this case, freedom from interference is required to allow coexistence of safety and non safety related components. For the protection of the Safety Transformers against failures from non-safety-related components, the Safety Transformers requires external protection mechanisms to achieve freedom from interference regarding memory. Failures that result from the Safety Transformers, are avoided by the development process of the Safety Transformers according to ASIL-D under the condition that the obligations that are documented in this safety manual are fulfilled.

6.2.1. Verifying the Software

6.2.1.1. Generic assumptions

[EB_E2ESE070004]

Verify that any other software that runs on the ECU is validated according to the safety integrity level of its partition.

[EB_E2ESE061004]

Verify that all transformers that are executed prior to the E2EXf have the safety level that is required by the system.

[EB_E2ESE061005]

Verify that all safety transformers that run in the same partition context as the E2EXf have the safety level that is required by the system.

[EB_E2ESE061028]

Verify that the caller invokes the read calls and write calls of the safety transformers periodically.
Note: In case the receiver is triggers periodically then timeout detection is ensured by evaluation the sequence counter. If the receiver is event-driven (i.e. receiver is triggered on the message reception), then the receiver shall implement its own timeout detection mechanism (is not part of the safety transformers).

[EB_E2ESE061029]

Verify that the following header files used directly or indirectly by the Safety Transformers complies with the requirements for the development of safety-related software for the automotive domain.

- ComXf_MemMap.h
- Std_Types.h
- TransformerTypes.h
- SchM_ComXfType.h
- SomeIpXf_MemMap.h
- SchM_SomeIpXfType.h
- stddef.h
- E2EXf_MemMap.h
- E2E_MemMap.h
- SCrc_MemMap.h

[EB_E2ESE061030]

Verify that the serializing transformer and the deserializing transformer (inverse transformer) have the same interpretation of the to be transformed data element.
Note: This can be ensured by performing a test that verifies the correct transmission of each data element.

[EB_E2ESE061031]

Verify that no C-identifier (i.e. an object; a function; a tag or a member of a structure, union, or enumeration; a typedef name; a label name; a macro name; or a macro parameter) is defined with a name that starts with "E2E", "COMXF", "SOMEIPXF" or "SCRC" (case insensitive) outside Safety Transformers.

6.2.1.2. Verifying the Rte

[EB_E2ESE061006]

Verify that the RTE does not interfere the safety-relevant data.

[EB_E2ESE061009]

Verify that the RTE invokes Safety Transformers to deserialize data in the `Rte_Read` or `Rte_Receive` function.

[EB_E2ESE061010]

Verify that the RTE invokes Safety Transformers to serialize data in the `Rte_Write` or `Rte_Send` function.

[EB_E2ESE061011]

Verify that the RTE must not invoke Safety Transformers if the previous transformer returns a hard error.

[EB_E2ESE061012]

Verify that the RTE invokes Safety Transformers if the previous transformer returns a soft error or no error.

[EB_E2ESE061013]

Verify that if `Rte_Read` or `Rte_Receive` is invoked and no new data since the last call is available, the RTE shall call the Safety Transformers as part of the inverse transformer chain using the last received data.

Note: In case of non queued reception and no new data is received then instead of message loss, unintended message repetition is reported. For queued reception, message loss is correctly reported.

[EB_E2ESE061014]

Verify that the RTE passes the return values of the Safety Transformers to the SWC.

[EB_E2ESE061015]

Verify that the RTE ensures that all buffers that are handed over to the Safety Transformers have a sufficient size.

[EB_E2ESE061016]

Verify that the RTE ensures that the buffer size indication that is passed to the Safety Transformers is correct.

6.2.2. Verifying the hardware

[EB_E2ESE070001]

Verify that the hardware provides a reliable execution environment. In the Safety Transformers context a reliable execution environment is defined as a microcontroller that provides mechanisms that prevent or detect non-systematic hardware faults, e.g. data corruption or incorrect execution of instructions. Examples are lockstep mode and ECC memory.

[EB_E2ESE065131]

Verify that the target MCU has to be identical to one of the reference platforms that are specified in [RELNOTES] or was approved separately by Elektrobit Automotive GmbH. If any other 16-bit or 32-bit target MCU is chosen, you must perform additional verification measures that are adequate and sufficient to ensure that there are no faults of the Safety Transformers that originate in the chosen target MCU.

[EB_E2ESE065132]

Verify that the item has to provide enough RAM, ROM, RunTime resources for *EB tresos Safety E2E Transformer*. For the calculation of the required resources, the values from the reference platforms that are specified in [RELNOTES] can be used as basis.

[EB_E2ESE070007]

Verify provisions against data errors in the Safety Transformers that are made, e.g. by the use of memory protection.

6.2.3. Verifying the software tools

[EB_E2ESE070002]

Verify that appropriate measures to ensure the confidence in the used software tools are taken, e.g. via tool qualifications. The used software tools include build environment, compiler, linker, and flash tool.

[EB_E2ESE070003]

Verify that the integrator reviews the output of each used tool is reviewed and appropriate measures are taken if warnings or errors exist. The used tools are the *Checker tool*, EB tresos Studio, the compiler, and the linker.

[EB_E2ESE065141]

Verify that the target software is delivered in source code and has requires an appropriate build environment, i.e. compiler, linker and flash tool to apply the created binary to the target hardware.

[EB_E2ESE065142]

Verify the compiler version and settings or options are identical to one of the reference platforms that are specified in [RELNOTES] or was approved separately by Elektrobit Automotive GmbH. If any other compiler, settings or options are chosen, perform adequate and sufficient verification measures to ensure that no faults of the *EB tresos Safety E2E Transformer* are caused by the chosen compiler, settings or options.

[EB_E2ESE065143]

Verify that the compiler does not report any errors. The compiler does not report any warnings or all generated warnings can be considered as acceptable, i.e. the code is correct for the used application, after they are carefully analyzed.

6.2.4. Verifying the communication description

[EB_E2ESE020224]

Verify that the communication is an explicit sender-receiver communication, in 1:1 and 1:N multiplicities.

[EB_E2ESE061027]

Verify that the `disableEndToEndCheck` parameter is set to `FALSE` in the system description.

[EB_E2ESE070009]

Verify that the *Communication description* for safety-relevant data elements includes the E2EXf in the *transformer chain*.

[EB_E2ESE070010]

Verify that `executeDespiteDataUnavailability` parameter is set to `TRUE` in the system description.

[EB_E2ESE070011]

Verify that a transformation with the same `transformerId` is not executed concurrently.
Note: The `transformerId` belongs to the `BswModuleEntry`.

For further information, see the module user's guide in the following documents:

- EB tresos E2E Transformer (E2E) documentation [E2EXFUG]
- EB tresos AutoCore Generic 7 Transformer (SOME/IP) documentation [SOMEIPUG]
- EB tresos AutoCore Generic 7 Transformer (COM) documentation [COMXFUG]
- EB tresos AutoCore Generic 8 Transformers documentation [ACGTRANSFORMERUG]

6.2.5. Verifying the application

[EB_E2ESE060001]

Verify that the selected E2E profile is appropriate for the indented use case.

[EB_E2ESE060004]

Verify that the chosen E2E profile with its constraints fulfills the safety goals regarding communication protection. Constrains are, e.g. maximum permitted length of the protected data to ensure an appropriate error detection capability. The transmission of one undetected erroneous data element in a sequence of data elements between sender and receiver does not lead to the violation of a safety goal of this system.

[EB_E2ESE061018]

Verify that the SWC does not interfere the safety-relevant data.

[EB_E2ESE061001]

Verify that the `E2EXf_<PartitionInfix>_PartitionInit()` API of the *Safety Transformers* is not interrupted by any other API function or by itself.

[EB_E2ESE061002]

Verify that the `E2EXf_<PartitionInfix>_PartitionInit()` API of the *Safety Transformers* does not interrupt any other API function.

[EB_E2ESE061003]

Verify that there is no unintentional call of the `E2EXf_<PartitionInfix>_PartitionInit()` API of the *Safety Transformers*.

[EB_E2ESE061020]

Verify that the corresponding initialization routine `ComXf_Init()` is not called from a safety related Software.
Note: The `ComXf_Init()` is part of the QM ComXf. The safe ComXf code has no Init function.

[EB_E2ESE061022]

Verify that for each partition, the corresponding initialization routine `E2EXf_<PartitionInfix>_PartitionInit()` is called before the first call of any `E2EXf_<transformerId>()` or `E2EXf_Inv_<transformerId>()` of the corresponding partition.

[EB_E2ESE061023]

Verify that the serializer transformer call `ComXf_<transformerId>()` is called before the `E2EXf_<transformerId>()` call.

[EB_E2ESE061024]

Verify that the transformer call `E2EXf_Inv_<transformerId>()` is called before the deserializer transformer call `ComXf_Inv_<transformerId>()`.

[EB_E2ESE061025]

Verify that the serializer transformer call `SomeIpXf_<transformerId>()` is called before the `E2EXf_<transformerId>()` call.

[EB_E2ESE061026]

Verify that the transformer call `E2EXf_Inv_<transformerId>()` is called before the deserializer transformer call `SomeIpXf_Inv_<transformerId>()`.

6.2.6. System assumptions

[EB_E2ESE060002]

Verify that the assumed E2E requirements (see Section 3.2.3, “Assumed safety requirements of Safety Transformers”) comply with the user's safety concept.

Appendix A. Safety Checklist

Phase	#	Activity	Chapter	QM	Safety	Applied [yes, no, partly]	Evidence/Justification/Comment
Delivery	1	Download, install	4.2. Installing Safety Transformers	x			
Delivery	2	Check delivery with check sum	4.3. Verifying the integrity of the Safety Transformers sources		x		
Delivery	3	Check delivery version, check that safety statement corresponds with the used version	2.3. Quality and safety statement		x		
Field Monitoring	4	Check list of published known issues	4.4. Field monitoring	x			
Technical Safety Concept	5	Check that assumed E2E Safety Transformer requirements are sufficient for the use case	3.2.3. Assumed safety requirements of Safety Transformers		x		
Technical Safety Concept	6	Check with the OEM that the selected E2E profile is appropriate for the use case.	Refer to the corresponding safety manual of the used profile		x		
Technical Safety Concept	7	Consider limitations of the Safety Transformers	3.2.12. Limitations of Safety Transformers	x			
Software Implementation/Integration	8	Consider the Workflow of the Safety Transformers	4.5.1. Workflow of the Safety Transformers		x		
Software Implementation/Integration	9	Consider assumed external functionality	5.1.3. Assumed external functionality				

Phase	#	Activity	Chapter	QM	Safety	Applied [yes, no, partly]	Evidence/Justification/Comment
Software Implementation/Integration	10	Configure the Safety Transformers	6.1. Configuration Rules	x			
Software Implementation/Integration	11	Check assumptions of the Safety Transformers	6.2. Assumptions of Safety Transformers		x		
Software Implementation/Integration	12	Execute the *Checker tool* for each used Transformer (SomeIpXf, ComXf and E2EXf)	Appendix B. Safety Checker		x		
System integration	13	Verify that the software requirements are fulfilled	6.2.1. Verifying the software		x		Implicitly covered by software Implementation/Integration phase checks
System integration	14	Verify that the hardware requirements are fulfilled	6.2.2. Verifying the hardware		x		Implicitly covered by software Implementation/Integration phase checks
System integration	15	Verify that the tool requirements are fulfilled	6.2.3. Verifying the software tools		x		Implicitly covered by software Implementation/Integration phase checks
System integration	16	Verify that the communication description requirements are fulfilled	6.2.4. Verifying the communication description	x			Implicitly covered by software Implementation/Integration phase checks
System integration	17	Check that application assumption are fulfilled	6.2.5. Application assumption	x			Implicitly covered by software Implementation/Integration phase checks
System integration	18	Check that system assumption are fulfilled	6.2.6. System assumption	x			Implicitly covered by software Implementa-

Phase	#	Activity	Chapter	QM	Safe-ty	Applied [yes, no, partly]	Evidence/Justifica-tion/Comment
							tion/Integration phase checks

Table A.1. Safety Checklist

Appendix B. Safety Checker

B.1. Safety Checker Usage

The Safety Transformers provide individual check tools checkE2EXf.bat, checkComXf.bat, and checkSomeIpXf.bat. Verify that the default relative path to the Java version provided by EB tresos Studio is set correctly in `setEnvVars.bat`.

The command line arguments of the check tools are shown in Figure B.1, “Check tool command line arguments”:

```
Options:
  -f, --file
    File or folder to check.
    Default: []
  -h, --help
    List of all available commands.
    Default: false
  -o, --output
    Output folder that will contain checker report.
    Default: .
  -v, --version
    Prints the version number of the checker tool.
    Default: false
```

Figure B.1. Check tool command line arguments

For the check of the files generated by E2EXF execute:
`checkE2EXf.bat -f E2EXf_Api.h -f E2EXf_Cfg.h -f E2EXf_[PARTITIONINFIX]_PartitionApi.h -f E2EXf_[PARTITIONINFIX]_PartitionApi.c`

For the check of the files generated by COMXF execute:
`checkComXf.bat -f ComXf_Api.h -f ComXf_Cfg.h -f ComXf_S_[PARTITIONINFIX]_PartitionApi.c -f ComXf_S_[PARTITIONINFIX]_PartitionApi.h`

For the check of the files generated by SOMEIPXF execute:
`checkSomeIpXf.bat -f SomeIpXf_Api_Gen.h -f SomeIpXf_Cfg.h -f SomeIpXf_Cfg.c -f SomeIpXf_S_[PARTITIONINFIX]_PartitionApi.h -f SomeIpXf_S_[PARTITIONINFIX]_PartitionApi.c`

In the following configuration cases no SomeIpXf_Cfg.c file is generated:

- only external trigger transformers are configured
- only transformers without actual data elements to serialize are configured (e.g. client server transformer without operation arguments)

therefore this file needs to be excluded from the above listed files.

For projects with mixed configurations only the safety relevant files can be verified by the checker. In case other files besides the ones listed above are used for verification, the checker will report an error.

As a first step the check tool verifies the integrity of itself and the imported files, if these checks have been performed successfully a Safety Check Report file is generated.

```
C:\Windows\system32\cmd.exe

C:\Projects\trunk\Tresos\tools\XfrmChecker>checkComXf.bat -v
Checking integrity of folder C:\Projects\trunk\Tresos\tools\XfrmChecker\.
checking integrity of file .\checkComXf.bat...
checking integrity of file .\checkE2EXf.bat...
checking integrity of file .\checkSomeIpXf.bat...
checking integrity of file .\checkXfrm.bat...
checking integrity of file .\md5deep.exe...
Checking integrity of folder C:\Projects\trunk\Tresos\tools\XfrmChecker\ComXfCU\.
checking integrity of file .\ComXfCheck.jar...
Error: Did not find hashsum of file .\ComXfCheck.jar.
Did not invoke checker since integrity check failed
C:\Projects\trunk\Tresos\tools\XfrmChecker>
```

Figure B.2. Example of failed integrity check due to mismatching hash sums.

B.2. Safety Check Report

The check tool creates as result a Check Report as XML file SafetyCheckReport.xml. The structure of the report is shown in Figure B.3, “Check Report.”.

```
  <?xml version="1.0" encoding="UTF-8"?>
- <report timestamp="YYYY/MM/DD hh:mm:ss" library_version="X.Y.Z" creator="eb.tresos.comxfcheck.Check vX.Y.Z" result="PASSED">
    + <input-files>
      <output-file>SafetyCheckReport.xml</output-file>
    + <review message="Verify that all listed partitions with its transformers and configurations are correct.">
    + <executed-checks>
  </report>
```

Figure B.3. Check Report.

The attributes of the XML tag `<report>` provide information regarding the creator of the report (version, name, and date). The overall check result is documented in the attribute `<result>` as "PASSED" or "FAILED".

The report sections `<review message>`, list manual review checks, that have to be performed by the user regaring the parameters of each transformer.

The last section of the report `<executed-checks>` provides information of the Check execution and has no relevance for the user.

Appendix C. Document configuration information

This document has been created by the DocBook engine using the source files and revisions listed below. All paths are relative to the directory https://subversion.ebgroup.elektrobit.com/svn/autosar/asc_E2ESE/asc_-E2ESEXfmgmt/stable/RFI_ACG-8.7.5_1/doc/SM.

Filename	Revision
../public/fragments/Bibliography.xml	4085
../public/fragments/Glossary.xml	3758
../SwAD/basic_software/interaction_view.xml	4188
../SwAD/E2EXf_autosar.xml	3883
../SwAD/Transformer.xml	3748
../SwRS/Failure_Model.xml	3748
../SwRS/TL-Requirements.xml	4239
Appendix_Safety_Checker.xml	4149
Appendix_Safety_Checklist.xml	4610
ASCE2ESE-519_Safety_E2E_Transformers_safety_manual.xml	3725
SM_About_the_Safety_E2E.xml	3988
SM_Application_constraints_and_requirements.xml	3089
SM_Configuration_verification_criteria.xml	4463
SM_Document_information.xml	3752
SM_History.xml	4610
SM_SEooC_Definition.xml	3752
SM_Using_the_Safety_E2E.xml	3990

Glossary

Application software	The application software is the software implemented within the AUTOSAR application layer. Among others this includes the *AUTOSAR application software components*.
Safety Transformer Checker	The Safety Transformers Checker is a tool to verify the generated code from SomeIpXf, ComXf, and E2EXf. In this way, the confidence in the generated code is increased. Each of the Transformers (SomeIpXf, ComXf, and E2EXf) has an own dedicated checker tool and this is included in the module plugin. ► The E2EXf checker tool verifies the generated code of the *E2EXf* module. ► The SomeIpXf checker tool verifies the generated code of the *SomeIpXf* module. ► The ComXf checker tool verifies the generated code of the *ComXf* module.
Com	AUTOSAR Com module.
Communication description	The communication description contains the information about the network, the communication matrix, and the protection of signals.
COM hardware	The hardware elements that are needed for the communication.
Physical COM link	The physical communication medium, i.e. wiring.
COM software	The AUTOSAR communication stack.
EB tresos Safety E2E Transformer (COM)	EB tresos Safety E2E Transformer (COM), short ComXf is a serializing transformer (signal based data serialization), that serializes data elements into linear arrays based on a fixed data mapping and vice versa. This byte array is processed by the E2EXf. The *Com* is used in the lower communication path. This allows using the serialization technology to ensure backward-compatibility on the network.
E2E	The E2E module is the end-to-end protection library, that includes the protection and the check routines.
E2ESM	The E2ESM implements a state machine to provide states and results to the consumer.
E2EXf	E2EXf calls the *E2E* library to either protect the serialized data stream in the sending path or check the received data stream for communication errors.
EB tresos Safety E2E Transformers	EB tresos Safety E2E Transformers, short Safety Transformers, is composed of the following products:

- *EB tresos Safety E2E Transformer (E2E)*
- *EB tresos Safety E2E Transformer (SOME/IP)*
- *EB tresos Safety E2E Transformer (COM)*

The end-to-end protected sending and receiving of data with an AUTOSAR stack. The EB tresos Safety E2E Transformers consists of an end-to-end communication protection library (*E2E*), the actual end-to-end communication transformer module (*E2EXf*), a module for data serialization and de-serialization, i.e. either *SomeIpXf* or *ComXf* , a state machine *E2ESM*, *SCrc* and different profiles, e.g. P01, P02, P04, P05, P06, and P07, for the integration into an AUTOSAR basic software stack.

`EB tresos Safety E2E Transformer (E2E)`

EB tresos Safety E2E Transformer (E2E), short E2EXf consists of the following modules:

- *E2EXf*
- *E2E*
- *SCrc*
- *E2ESM*

`Electronic control unit (ECU)`

An AUTOSAR ECU, compare with AUTOSAR Glossary.

`Integrator`

A person who integrates the software on the ECU.

`LdCom`

The AUTOSAR LdCom (Efficient COM for Large Data) module is an alternative for the Com module. The LdCom does not perform any data serialization.

`Rte`

The Runtime Environment or AUTOSAR Runtime Environment is an AUTOSAR module.

`Safety RTE`

Safety RTE is a safety qualified version of the *Rte*.

`SCrc`

The SCrc module is the safety implementation of the CRC.

`Software component (SWC)`

An AUTOSAR software component.

`EB tresos Safety E2E Transformer (SOME/IP)`

EB tresos Safety E2E Transformer (SOME/IP), short SomeIpXf transformer is a serializing transformer (generic data serialization format), that serializes data elements into linear arrays of dynamic size and vice versa. This byte array is processed by the E2EXf. The *Com* and the *LdCom* , which does not perform any data serialization) can be used for efficient communication.

EB tresos Studio	Provides a GUI to administrate and control target software components. EB tresos Studio is invoked by the *Integrator* and provides an interface to import the AUTOSAR *communication description* into an internal database that can be accessed by EB tresos Studio public APIs.
System description	A system description is an AUTOSAR system description.
EB tresos Safety TimE Protection	EB tresos Safety TimE Protection is a safety-qualified module for temporal and logical program flow monitoring.
Transformation	Conversion of a single data element of any complexity with a single *transformation technology*, e.g. serializing of data.
Transformation Chain	Two or more *Transformations* in a row.
Transformation Technology	Implementation of a component that allows to convert a data elements from one representation into another.

Bibliography

[ACGTRANS-FORMERUG] *EB tresos AutoCore Generic 8 Transformers documentation*:
AutoCore_Generic_Transformers_documentation.pdf

[ASRSWSE2E] *Specification of SW-C End-to-End Communication Protection Library V4.2.1, R4.2 Rev 1, 2012*, 2012

[COMXFUG] *EB tresos AutoCore Generic 7 Transformer (COM) documentation*:
AutoCore_Generic_COMXF_documentation.pdf

[E2EXFUG] *EB tresos E2E Transformer (E2E) documentation*:
E2E_E2EXF_documentation.pdf

[ISO26262_1ST] *INTERNATIONAL STANDARD ISO 26262*: Road vehicles - Functional safety, 2011
http://qms/portal/1.0.1/release/other_content/guidances/examples/Functional%20Safety_4B52FF3E.html

[ISO26262-1_1ST] *INTERNATIONAL STANDARD ISO 26262-1*: Road vehicles - Functional safety - Part 1: Vocabulary, 2011
http://qms/portal/1.0.1/release/other_content/guidances/examples/resources/ISO_26262-1_%28E%29_2011-11.pdf

[ISO26262-10_DIS] *INTERNATIONAL STANDARD ISO 26262-10*: Road vehicles - Functional safety - Part 10: Guideline on ISO 26262 analyses, 2011
http://qms/portal/1.0.1/release/other_content/guidances/examples/resources/ISO_26262-10_(E)_2011-01.pdf

[RELNOTES] *Release Notes*
http://subversion.ebgroup.elektrobit.com/svn/autosar/asc_E2ESE/asc_E2ESEXfmgmt/trunk/doc/RN/release/ASCE2ESE-623-Safety_E2E_Transformers_release_notes.pdf

[RFC_2119] *IETF RFC 2119 (Key words for use in RFCs to Indicate Requirement Levels), March 1997,*
http://www.ietf.org/rfc/rfc2119.txt

[SOMEIPUG] *EB tresos AutoCore Generic 7 Transformer (SOME/IP) documentation*:

AutoCore_Generic_SOMEIPXF_documentation.pdf