Document Title	Specification of Safety Extensions
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	671

Document Status	Final
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	4.3.1

Document Change History			
Date	**Release**	**Changed by**	**Description**
2017-12-08	4.3.1	AUTOSAR Release Management	minor corrections / clarifications / editorial changes; For details please refer to the ChangeDocumentation
2016-11-30	4.3.0	AUTOSAR Release Management	improved modeling of decomposition relation of safety requirements; minor corrections / clarifications / editorial changes; For details please refer to the ChangeDocumentation
2015-07-31	4.2.2	AUTOSAR Release Management	minor corrections / clarifications / editorial changes; For details please refer to the ChangeDocumentation
2014-10-31	4.2.1	AUTOSAR Release Management	Initial specification based on Concept "'Safety Extensions"'

Disclaimer

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

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

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

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

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

Table of Contents

Bibliography

[1] Standardization Template
AUTOSAR_TPS_StandardizationTemplate

[2] Requirements on Safety Extensions
AUTOSAR_RS_SafetyExtensions

[3] ISO 26262 (Part 1-10) – Road vehicles – Functional Safety, First edition
http://www.iso.org

[4] Technical Safety Concept Status Report
AUTOSAR_TR_SafetyConceptStatusReport

[5] Methodology
AUTOSAR_TR_Methodology

0.1 Document Conventions

The representation of requirements in AUTOSAR documents follows the table specified in [TPS_STDT_00078], see Standardization Template, chapter Support for Traceability ([1]).

The verbal forms for the expression of obligation specified in [TPS_STDT_00053] shall be used to indicate requirements, see Standardization Template, chapter Support for Traceability ([1]).

1 Introduction

1.1 Overview

This document contains the specification of the AUTOSAR Safety Extensions and realizes the requirements stated in [2]. Safety extensions are expressed by existing (generic) AUTOSAR meta-model concepts. Native meta-model concepts might be introduced in upcoming releases. Section 3 provides a more detailed overview on the extensions.

1.2 Scope

The scope of this document covers safety extensions that shall enable ISO 26262 development in an AUTOSAR context. These extensions allow a standardized exchange of safety information and provide the basis for consistent management among different vendors and tools as required by ISO 26262.

This document is not an introduction to functional safety in general or ISO 26262 in specific. Other safety standards or guidelines such as IEC 61508 or MISRA are out of scope.

The referenced deliverable TR SafetyConceptStatusReport is set to status obsolete in release 4.3.1.

1.3 Abbreviations

Abbreviation	*Meaning*
ASIL	Automotive Safety Integrity Level
DC	Diagnostic Coverage
ECC	Error Correction Code
EDC	Error Detection Code
HARA	Hazard Analysis and Risk Assessment
HW	Hardware
FSC	Functional Safety Concept
TSC	Technical Safety Concept
SEooC	Safety Element out of Context
SM	Safety Mechanism or Measure
SW	Software
SWC	Software Component
URI	Uniform Resource Identifier
URL	Uniform Resource Locator

Table 1.1: Abbreviations

1.4 Glossary of Terms

In general this document will use terms related to safety as defined in ISO 26262-1, *Vocabulary* (see [3]). For clarification table 1.2 lists some terms with definitions in relationship to AUTOSAR.

Term	*Definition*
ASIL attribute	The ASIL for elements of the system specify the necessary requirements of ISO 26262 and **safety measures** to apply for avoiding unreasonable residual risk. See section 5 for further details.
Fault, Failure, Error	A **fault** is an abnormal condition that *may* cause an HW or SW element to fail. An **error** describes the resulting discrepancy in a value or condition and is the consequence of a (set of) faults. A **failure** defines the termination of the ability of an HW or SW element to perform its function (see [3]). Faults comprise systematic SW faults (i.e. ”defects”, ”bugs”), random HW faults (e.g. due to stress/aging of the equipment) as well as systematic HW faults.
Safe state	A safe state is always meant to be described on system level (see [3]). A certain software state *may* be part of this ”system state” or the relation might be undefined (e.g. if the microcontroller running the software is switched off in the safe state).
Safety Mechanism	A safety mechanism is a *technical solution [...], to detect faults or control failures in order to achieve or maintain a safe state* (see [3]). The term is used in this specification exactly in this broader sense, so that not only the AUTOSAR safety mechanisms (”safety features”) can be described, but any HW/SW or combined solution of the system for which an AUTOSAR software is implemented (cp. section 7).
Safety Measure	A safety measure is an *activity or solution to avoid systematic failures and to detect random hardware failures or control failures* (see [3]). Therefore, a safety measure might only define a process activity like dedicated testing methods, additional code verifications, and so on (cp. section 7). This specification will use the term safety measure to subsume both activities during development as well as safety measure implemented into the system.
Safety Requirement	ISO 26262 defines a hierarchy of safety requirements: safety goals, technical, hardware and software. In this document a safety requirement could be any of these. For details refer to ISO 26262-3, 4 and 9.

Table 1.2: Glossary of terms

1.5 Guidelines

Existing specifications shall be referenced (in form of a single requirement). Differences to these specifications are specified as additional requirements. All Requirements shall have the following properties:

- Redundancy
  Requirements shall not be repeated within one requirement or in other requirements.
- Clearness
  All requirements shall allow one possibility of interpretation only. Used technical terms that are not in the glossary must be defined.
- Atomicity
  Each Requirement shall only contain one requirement. A Requirement is atomic if it cannot be split up in further requirements.
- Testability
  Requirements shall be testable by analysis, review or test.
- Traceability
  The source and status of a requirement shall be visible at all times.

2 Requirements Tracing

The following table references the requirements specified in [2] and links to the fulfillments of these.

Requirement	Description	Satisfied by
[RS_SAFEX_00001]	Safety Requirements expressible within AUTOSAR Models	[TPS_SAFEX_00101]
[RS_SAFEX_00002]	Safety Requirements at least as expressive as other Requirements	[TPS_SAFEX_00101]
[RS_SAFEX_00003]	Safety Requirements Description by an URI	[TPS_SAFEX_00105]
[RS_SAFEX_00004]	Safety Requirements distinguishable	[TPS_SAFEX_00102]
[RS_SAFEX_00005]	Safety Requirements uniquely identifiable	[TPS_SAFEX_00103]
[RS_SAFEX_00006]	Status Information for Safety Requirements	[TPS_SAFEX_00104]
[RS_SAFEX_00007]	Hierarchy of Safety Requirements	[TPS_SAFEX_00301]
[RS_SAFEX_00008]	Decomposition of Safety Requirements	[TPS_SAFEX_00302]
[RS_SAFEX_00009]	Specification of Independence Requirements	[TPS_SAFEX_00303]
[RS_SAFEX_00010]	ASIL Attribute for Safety Requirements	[TPS_SAFEX_00201]
[RS_SAFEX_00011]	ASIL Attribute for AUTOSAR Elements	[TPS_SAFEX_00202]
[RS_SAFEX_00012]	Safety Requirements traceability	[TPS_SAFEX_00101]
[RS_SAFEX_00013]	Safety Measures traceability	[TPS_SAFEX_00401]
[RS_SAFEX_00014]	Safety Requirements Allocation	[TPS_SAFEX_00306] [TPS_SAFEX_00308]
[RS_SAFEX_00015]	Safety Measures expressible within AUTOSAR Models	[TPS_SAFEX_00401]
[RS_SAFEX_00016]	Textual Description of Safety Measures	[TPS_SAFEX_00401]
[RS_SAFEX_00017]	Safety Measures uniquely identifiable	[TPS_SAFEX_00402]
[RS_SAFEX_00018]	Relation between Safety Requirements and Safety Measures	[TPS_SAFEX_00307]
[RS_SAFEX_00022]	Safety Measures Allocation	[TPS_SAFEX_00305] [TPS_SAFEX_00309]
[RS_SAFEX_00023]	Safety Mechanisms as special Safety Measures	[TPS_SAFEX_00401]

3 Safety Extensions Overview

Safety is one of the key issues in automotive system design and development. ISO 26262 [3] defines the current standard for functional safety which impacts almost all development activities, including software specifications, design and implementation. This document enables a standardized exchange of this safety information in an AUTOSAR context and provide the basis for consistent management as required by ISO 26262.

The AUTOSAR standard addresses functional safety already by providing a number of features that can be facilitated to implement safe software, for example end to end protection, program flow monitoring, memory partitioning, user/supervisor-modes, and so on (see [4] for an overview). These *safety mechanisms* are recognized as one integral part of an AUTOSAR system design. However, additional requirements from ISO 26262 for functional safety software development need to be addressed, especially the following:

- Safety requirements – clearly distinguishable from other requirements and fulfilling the needs as specified by ISO 26262 parts 4 and 8 (section 4),
- Safety integrity levels – for each AUTOSAR element following the schema of ISO 26262-3 (section 5),
- Decomposition of safety requirements according to the needs as given in ISO 26262-9 (section 6)
- Traceability and allocation of safety requirements and safety measures according to ISO 26262 parts 4, 6 and 8 (section 6), and
- Safety measures and safety mechanisms as required by ISO 26262-4 (section 7). This goes beyond the pure SW safety mechanisms that exist in AUTOSAR and introduces an abstract way to reference any safety measure of a system architecture.

This specification follows the approach to reuse available documentation capabilities of AUTOSAR to address these requirements. This means that the `Safety Extensions` define rules to exchange the aforementioned work products by using existing meta-model concepts (e.g. `StructuredReq`, `TraceableText`, `trace`). Thereby, specifications of AUTOSAR remain backward compatible and can contain at the same time unified and tool-processable safety information for the development of safe SWCs and configurations (cp. RS-SafetyExtensions requirements [RS_SAFEX_00020] and [RS_SAFEX_00021]).

The hierarchy of safety requirements for a system (*item* development in ISO 26262) and its relation to an AUTOSAR software architecture is depicted in figure 3.1. The hierarchy of safety requirements starts with safety goals that are identified for the hazards/hazardous events of the system. The ASIL is maintained as attribute at each safety goal and inherited consistently through the subsequent levels of functional safety re-

quirements (as part of the FSC) and technical safety requirements (as part of the TSC). The latter will be refined into SW and HW safety requirements.

Each safety requirement[1] must be allocated properly to an element of the system architecture, i.e. component, HW, SW or both (HW and SW). Hence, an element of an AUTOSAR specification might receive an ASIL which indicates that it is in the scope of an ISO 26262 development.

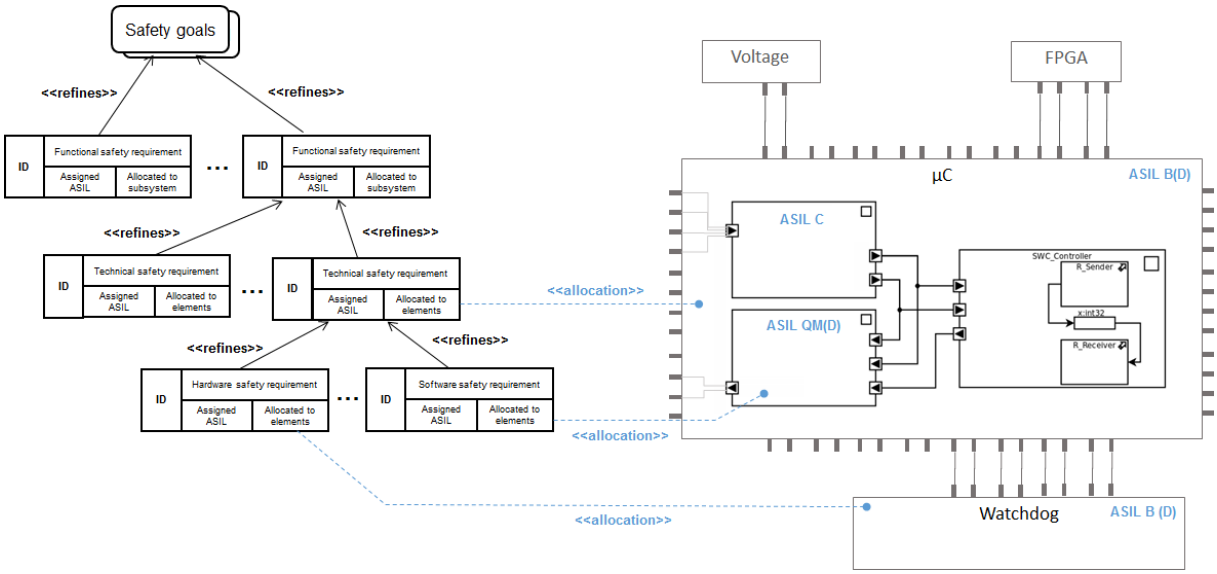

Figure 3.1: Hierarchy of safety requirements and allocation to system architecture elements

In cases where safety requirements are not available or will not be exchanged together with a specification, the AUTOSAR implementation must at least be *aware* that the element is used in a safety context. This is achieved by attaching the ASIL attribute to AUTOSAR elements independent from the allocation. Especially in cases of an SEooC development, where the safety requirements are not fully known at development time, the ASIL attribute supports the integration and verification of such parts in a later stage of development by matching the assumptions against the finalized safety requirements.

From the perspective of an AUTOSAR element the realization of allocated safety requirements is often dependent on the system context. For example, an implementer of a SWC shall be aware whether there is a memory protection (e.g by ECC/EDC/MMU/MPU) supported by the underlying processor architecture in order to correctly implement the handling of safety related data. Especially decomposition and allocation of safety requirements to other elements of the architecture — as well as constraints and characteristics of supporting parts — need to be known during development time. This is generally the case for most error detection and error handling, degradation or timing aspects. For example, the system excerpt in figure 3.1 indicates the availability of an external HW watchdog that might be a supportive element in the error handling procedures (e.g. deadline or output monitoring). The example application software

[1] Functional safety requirements are allocated to a higher-level functional/logical architecture

might rely on this *safety mechanism* for certain failures that cannot be detected by the component itself.

In order to convey the relevant information of this ”safety context” for development, integration, and configuration of AUTOSAR software this specification provides an abstraction of *safety measure* or *safety mechanism* in addition to safety requirements. Figure 3.2 shows the concept of the abstraction of different safety mechanisms available in the software stack and/or ECU hardware.

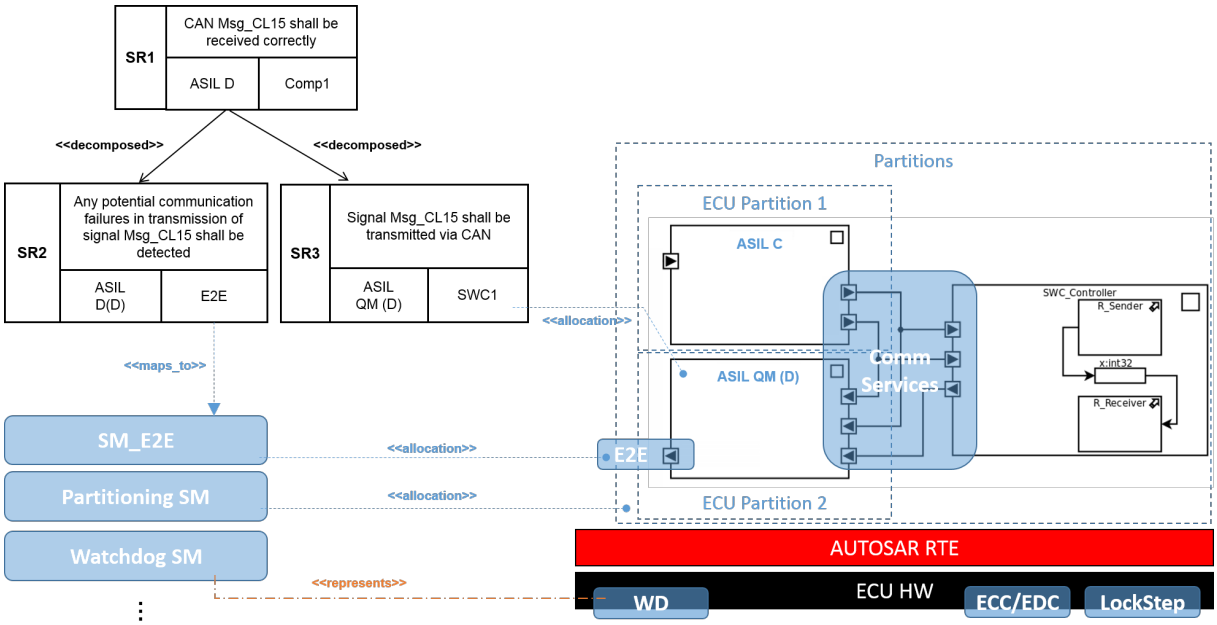

Figure 3.2: Safety measures, safety requirements and allocations to elements of the architecture

As shown in the figure a (decomposed) safety requirement is first mapped to an abstract definition of a safety mechanism (here: *SM_E2E*). In a subsequent step the safety mechanism is allocated to certain elements of an AUTOSAR model. In case the safety mechanism represents any other technology, this allocation is only implicit (not part of AUTOSAR). This allows for example the system integrator to verify whether freedom from interference in a decomposition is sufficiently achieved across the different technologies. Note that this abstraction is also useful, e.g. if the AUTOSAR (implementation) elements are not yet available in a distributed work between OEM supplier, but the system engineer wants to determine already what aspects are protected in which way by safety measures.

The individual activities of defining safety requirements or safety measures, allocating safety requirements, and so on is described in the AUTOSAR Methodology (cp. [5]). Hence, the Methodology addresses formally requirement [RS_SAFEX_00024] of [2]

4 Safety Requirements

This chapter defines how safety requirements will be mapped to AUTOSAR concepts. Basically, safety requirements follow the same rationale as normal requirements, but must fulfill additional criteria to meet ISO 26262 needs (cp. [3], part 8, clause 6.4.2). This comprises mainly additional attributes and characteristics that are addressed in AUTOSAR as follows:

- Safety requirements shall be unambiguously identifiable as safety requirements (see ISO 26262-8, clause 6.4.2.1). For this purpose requirements are tagged by the `category` attribute inherited by `StructuredReq`.
- Allocation information of safety requirements to elements of the (software) architecture shall be available (see ISO 26262-8, clause 6.4.2.3). Safety requirements are mapped to (any) object of the AUTOSAR architecture by means of a `trace`.
- Safety requirements shall have a unique identification that remains unchanged throughout the life-cycle of the requirement (ISO 26262-8, clause 6.4.2.5.a). This specification will make additional requirements on `shortName` usage for safety requirements.
- Safety requirements shall have a status attribute (ISO 26262-8, clause 6.4.2.5.b). The status attribute is different from the AUTOSAR lifecycle information defined for requirements and hence it is mapped to a `Sdg` property.
- Safety requirements shall have an ASIL (ISO 26262-8, clause 6.4.2.5.c). The ASIL attribute is mapped to a `Sdg` property.
- Safety requirements shall be structured hierarchically along design levels and each shall maintain a reference to the source at the upper level of the hierarchy (ISO 26262-8, clause 6.4.3.1 and clause 6.4.3.2). Since AUTOSAR allows the tracing of requirements as `TraceableText` there is no extension required for expressing these hierarchical dependencies.
- If ASIL decomposition is applied, the decomposition must follow a number of rules that are defined in ISO 26262-9, clause 5. This specification introduces a special trace type that supports the concept of ASIL decomposition individually on each safety requirement. Moreover, the ASIL decomposition notation is supported at the ASIL attribute, for example `ASIL B(D)`.

[TPS_SAFEX_00101] Description of safety requirements ⌈ Safety requirements shall be described as normal requirements using `StructuredReq` as defined in [TPS_STDT_00060]. The `description` shall contain the contents of the requirements. ⌋*(RS_SAFEX_00001, RS_SAFEX_00002, RS_SAFEX_00012)*

Note that this integrates seamlessly in the traceability of text defined for AUTOSAR specifications.

[TPS_SAFEX_00103] Unique identifier of safety requirements ⌈ Safety Requirements shall receive a unique ID across the extent of an AUTOSAR project. The ID

shall be maintained as shortName for further references to the requirement and correspond to the general rule [TPS_GST_00021]. ⌋(*RS_SAFEX_00005*)

Note that safety requirement identifiers are thus much stricter than normal short names defined by [constr_2508]. The shortName is used as a global unique ID, which is similar to the uniqueness of other elements as described in [constr_2538]. In addition, however, tools processing the safety extensions can facilitate the uuid attribute to persist tool related identifiers.

[TPS_SAFEX_00102] Type of safety requirements ⌈ Safety Requirements shall be marked unambiguously as safety requirement by the category attribute of StructuredReq set to one of the following:

- SAFETY_GOAL
- SAFETY_FUNCTIONAL
- SAFETY_TECHNICAL
- SAFETY_SOFTWARE
- SAFETY_HARDWARE
- SAFETY_EXTERNAL

These values extend the defined values in [constr_2540] in [1] in the context of safety. ⌋(*RS_SAFEX_00004*)

The ASIL attribute is defined in [TPS_SAFEX_00201].

[TPS_SAFEX_00104] Status attribute ⌈ Safety Requirements shall receive a status attribute as AdminData containing a Sdg data field with gid="SAFEX". The XML contents shall contain an Sd element with attribute gid="STATUS". ⌋(*RS_SAFEX_00006*)

The values of the status attribute are not prescribed and implementation specific.

For various reasons, it is not practicable to exchange a whole hierarchy of safety requirements inside the scope of an AUTOSAR project and/or a set of AUTOSAR XML documents. For example, the reference to HW safety requirements or safety goals might be deliberately excluded or safety requirements might be reside in a requirements database. In order to support the linking of such safety requirements that reside outside of AUTOSAR, this specification introduces the concept of External Safety Requirements.

[TPS_SAFEX_00105] External Safety Requirements ⌈ An External Safety Requirement that shall be included *as reference* in an AUTOSAR document shall be marked with a category set to SAFETY_EXTERNAL and the description shall contain only an Xfile URI to the location where the safety requirement resides. ⌋ (*RS_SAFEX_00003*)

Optionally the ASIL and/or status attributes might be set (as cache) for convenience as defined in [TPS_SAFEX_00201] and [TPS_SAFEX_00104] as well as the tool and toolVersion.

The listing below shows an example how safety requirements shall be expressed in AUTOSAR XML (Note:This listing contains elements resulting from specification items introduced in later sections of this document):

Listing 4.1: AUTOSAR XML representation of a safety requirement

```
<!-- A technical safety requirement -->
<STRUCTURED-REQ>
  <SHORT-NAME>SysSafReq05</SHORT-NAME>
  <LONG-NAME>
    <L-4 L="EN">CL15_ON light switch HW lib</L-4>
  </LONG-NAME>
  <CATEGORY>SAFETY_TECHNICAL</CATEGORY>
  <ADMIN-DATA>
    <SDGS>
      <SDG GID="SAFEX">
        <SD GID="ASIL">B</SD>
        <SD GID="STATUS">PROPOSED</SD>
      </SDG>
    </SDGS>
  </ADMIN-DATA>
  <TRACE-REFS>
    <!-- Traceability link to upper hierarchy (here: functional safety
       requirement) -->
    <TRACE-REF DEST="STRUCTURED-REQ" BASE="SAFEX">FSR02</TRACE-REF>
  </TRACE-REFS>
  <TYPE>Valid</TYPE>
  <DESCRIPTION>
    <P>
      <L-1 L="EN">While CL15ON==1 FLM ECU shall switch the light off only
        if HW_LB ==1 condition is true continuously for 20 ms. (CAN-
        message:  CL15_01 CAN-Signal: CL15ON Boolean, '1' if clamp 15 is
        set to on '0' if clamp 15 is set to off).</L-1>
    </P>
  </DESCRIPTION>
  <RATIONALE />
  <DEPENDENCIES />
  <USE-CASE />
  <SUPPORTING-MATERIAL />
</STRUCTURED-REQ>

<!-- An external technical safety requirement -->
<STRUCTURED-REQ>
  <SHORT-NAME>SysSafReq42</SHORT-NAME>
  <LONG-NAME>
    <L-4 L="EN"></L-4>
  </LONG-NAME>
  <CATEGORY>SAFETY_EXTERNAL</CATEGORY>
  <ADMIN-DATA>
    <SDGS>
      <SDG GID="SAFEX">
        <SD GID="ASIL">C</SD>
        <SD GID="STATUS">ACCEPTED</SD>
      </SDG>
    </SDGS>
  </ADMIN-DATA>
```

```
  <TRACE-REFS>
    <TRACE-REF DEST="STRUCTURED-REQ" BASE="SAFEX">FSR02</TRACE-REF>
  </TRACE-REFS>
  <TYPE>Valid</TYPE>
  <DESCRIPTION>
    <P>
      <L-1 L="FOR-ALL">
        <XFILE>
          <SHORT-NAME>SysSafReq42</SHORT-NAME>
          <URL>http://requirements.mycompany.com:6777/db/prj/safety/
              SysSafReq42</URL>
          <TOOL>My Requirements Tool</TOOL>
          <TOOL-VERSION>9.3.1</TOOL-VERSION>
        </XFILE>
      </L-1>
    </P>
  </DESCRIPTION>
  <RATIONALE />
  <DEPENDENCIES />
  <USE-CASE />
  <SUPPORTING-MATERIAL/>
</STRUCTURED-REQ>
[...]
```

5 Safety Integrity Levels

This specification is intended to support the Automotive Safety Integrity Level (ASIL) of ISO 26262 [3]. Other safety integrity levels will not be considered and are out of scope of this document.

The ASIL is determined as part of the HARA in the concept phase as of ISO 26262-3 and assigned to each safety goal. A system design – and finally the software architecture – will inherited this ASIL as an attribute via the allocation of safety requirements to the technical/software architecture (cp. section 3, see section 6 for allocation of safety requirements).

[TPS_SAFEX_00201] ASIL attribute of safety requirements ⌈ Safety requirements defined according to section 4 shall receive an ASIL attribute. The ASIL is stored at an `AdminData` that contains a `Sdg` data with `gid="SAFEX"`. The contents of this element shall contain an `Sd` element with attribute `gid="ASIL"`. Valid values for this attribute are:

- `QM`
- `A`
- `B`
- `C`
- `D`
- `QM(A)`
- `QM(B)`
- `QM(C)`
- `QM(D)`
- `A(B)`
- `A(C)`
- `A(D)`
- `B(B)`
- `B(C)`
- `B(D)`
- `C(C)`
- `C(D)`
- `D(D)`

⌋*(RS_SAFEX_00010)*

Note that the parentheses notation is used to express decomposed safety requirements. In this specification we will refer to the original ASIL (i.e. the value in parentheses) as the *contextual ASIL* before decomposition, since it belongs to the context of safety goal.

[constr_6200] Safety goals have no decomposed ASIL ⌈ If a safety requirement is of type `SAFETY_GOAL` the valid values of the `ASIL` attribute are restricted to: `QM, A, B, C,` or `D`. ⌋*()*

[TPS_SAFEX_00202] ASIL for AUTOSAR elements (optional) ⌈ All AUTOSAR elements should receive an `ASIL` attribute, if at least one safety requirement is allocated to it. The ASIL shall be added as `Sdg` data with `gid="SAFEX"` to the `AdminData` section in XML. The XML contents shall contain an `Sd` element with attribute `gid="ASIL"`, valid values are the same as in [TPS_SAFEX_00201]. ⌋*(RS_SAFEX_00011)*

Note that the ASIL at an element according to [TPS_SAFEX_00202] is optional.[1] If the ASIL is not specified at the element, the semantics is that it is derived as highest ASIL from all of the allocated safety requirements.

[constr_6201] Consistency of ASIL values ⌈ The ASIL of AUTOSAR elements and allocated safety requirements should be *consistent*. An ASIL is consistent if the value at an element is the same or higher of the maximum ASIL of allocated safety requirements. ⌋*()*

Note that an ASIL of an AUTOSAR element might be higher than the ASIL of safety requirements for various reasons. For example, a SWC might be designed for reuse in higher safety integrity contexts and therefore be rated with higher ASIL. For decomposed requirements, however, it is open to interpretation how the contextual ASIL is considered in the comparison of ASIL values.

For an example of the ASIL attribute at safety requirements see listing 4.1.

Listing 5.1: Example for the AUTOSAR XML representation of an ASIL attribute at an element

```
<!-- Example AUTOSAR element with ASIL -->
<APPLICATION-SW-COMPONENT-TYPE>
  <SHORT-NAME>MyComponent</SHORT-NAME>
    <ADMIN-DATA>
      <SDGS>
        <SDG GID="SAFEX">
        <SD GID="ASIL">B</SD>
      </SDGS>
    </ADMIN-DATA>
  <PORTS>
[...]
```

[1] This might be useful in an SEooC or carry-over development, where an existing specification is connected to safety requirements after implementation

6 Safety Requirements Traceability and Allocation

The essential characteristics of safety requirements according to ISO 26262 is the management and maintenance of traceability. This specification refers to *traceability of safety requirements* as the generic term for different types of links between (safety) requirements and other elements. Mainly three types of traces are distinguished:

1. *Refinement relations* between two levels of safety requirements, e.g. technical safety requirements that contribute to functional safety requirements (see ISO 26262-8, clause 6.4.3.1.a). This concept is similar to the upstream tracing of the AUTOSAR specification itself and will be realized in the same way.
2. *Allocation relations* from safety requirements to software architecture elements, e.g. a SW safety requirement allocated to a port of an AUTOSAR SWC (see ISO 26262-8, clause 6.4.2.3).
3. *Mapping relation* from safety requirements to safety measures/mechanisms, e.g. a safety requirement for a CRC that is mapped to an end to end protection safety mechanism (see ISO 26262-4, clauses 6.4.1, 6.4.2, and 6.4.6).

Note that traceability of safety requirements do not solely refer to references between text elements as in the current AUTOSAR documentation meta-model (see [TPS_GST_00243]). Therefore, the different relation types are managed in `AdminData` blocks using a `Referrable` reference (via `sdx` elements).

Decomposition is a specialization of the refinement relation that has architectural implications. A decomposition of a safety requirement requires two *independent* elements in the system architecture to exist, for which freedom from interference can be guaranteed. In order to trace these decompositions via decomposed safety requirements down to software, we are increasing the awareness of implementors and enable verification of the same e.g. during integration testing.

[TPS_SAFEX_00301] Safety requirement refinement relations ⌈ Refinement relations of safety requirements shall be expressed by `trace` associations. The direction of the trace has the semantic ”refines”. ⌋*(RS_SAFEX_00007)*

[TPS_SAFEX_00302] Decomposition of safety requirements ⌈ Decomposition shall be specified at each of the two decomposing requirements into which a safety requirement is decomposed. For this purpose both of these decomposing requirements shall receive an `AdminData` entry containing a `Sdg` element named `gid=”DECOMPOSITION”` that has a reference as `sdx` (i.e. `Referrable`) to the decomposed safety requirement. ⌋*(RS_SAFEX_00008)*

[constr_6202] Decomposition into two safety requirements ⌈ A decomposition as specified by [TPS_SAFEX_00302] shall be specified at exactly two decomposing safety requirements (not more) for each decomposed requirement. ⌋*()*

[constr_6203] Decomposing only one safety requirement ⌈ Each decomposing requirement specified according to [TPS_SAFEX_00302] shall decompose maximum one other requirement. ⌋*()*

[TPS_SAFEX_00303] Independence requirement link ⌈ If safety requirements express a means to achieve freedom from interference for elements of a decomposition, they shall be listed in addition to the decomposed requirement at both of the decomposing safety requirements. Therefore the AdminData of each of the decomposing safety requirements receives a separate reference (sdx entry) in an Sdg element with gid="INDEPENDENCE". ⌋*(RS_SAFEX_00009)*

Note that the decomposed safety requirement and the requirement for independence may receive in addition the "reversed" trace to the decomposing safety requirements. In that way the whole traceability hierarchy can be navigated seamlessly by tools that are not aware of safety extensions.

[TPS_SAFEX_00306] Allocation of safety requirements to AUTOSAR elements ⌈ Allocation of a safety requirement to AUTOSAR elements is expressed by means of a reference in the AdminData block that points to the AUTOSAR element. For each allocation reference, a sdx reference shall be listed in a combined Sdg element named gid="ALLOCATION". ⌋*(RS_SAFEX_00014)*

An alternative to the direct allocation of safety requirements to AUTOSAR elements is first a *mapping* to safety measures (if applicable) and subsequently to AUTOSAR elements. For example, a safety requirement to ensure safe communication could be mapped to a safety mechanism "End to End Protection" which in turn is allocated to an end to end profile.

[TPS_SAFEX_00305] Mapping of safety requirements to safety measures ⌈ Allocation of a safety requirement to a safety measure shall be mapped to a sdx reference in an Sdg element (in AdminData block) with name gid="MAPS_TO" containing sdx references to the safety measure(s). ⌋*(RS_SAFEX_00022)*

As fully equivalent alternative the safety mechanism may contain a backward link to the safety requirements it will realize:

[TPS_SAFEX_00309] Alternative relationship of the mapping relation ⌈ Mapping relations shall be expressed by a trace associations from the safety measure to the safety requirement. The direction of the trace has the semantic "realizes". ⌋ *(RS_SAFEX_00022)*

[TPS_SAFEX_00307] Allocation of safety measures to AUTOSAR elements ⌈ The mapping of a safety measure to one (or more) AUTOSAR element(s) shall be expressed in AdminData containing a Sdg named gid="ALLOCATION", containing sdx references to the AUTOSAR elements. ⌋*(RS_SAFEX_00018)*

From the perspective of an AUTOSAR element the allocation links have a *realizes* (or *satisfies*) semantics: the element has to implement all of the allocated safety requirements and defined safety mechanisms. Therefore this specification provides a fully equivalent alternative to these relations by means of a realizes relationship:[1]

[1]This might be useful in cases where the (safety) requirements specification is baselined and should not be changed

[TPS_SAFEX_00308] `Realizes` relationship of AUTOSAR elements ⌈ The allocation of safety requirements or safety measures may be expressed by a `realizes` reference. The reference shall be added as `Sdg` data to the `AdminData` section of the element with attribute `gid="REALIZES"`. The XML contents shall contain an `Sd` element with a list of `sdx` references referring to the allocated safety requirements. ⌋ *(RS_SAFEX_00014)*

Listing 6.1: Example for the AUTOSAR XML representation of realizes relationship

```
[...]
    <AR-PACKAGE>
  <SHORT-NAME>FLM_swc</SHORT-NAME>
      <ELEMENTS>
        <APPLICATION-SW-COMPONENT-TYPE>
          <SHORT-NAME>FLM</SHORT-NAME>
    <ADMIN-DATA>
    <SDGS>
      <SDG GID="ASIL">
        <SD>B</SD>
      </SDG>
      <!-- Example showing the <<realizes>> relationship (cp.
          TPS_SAFEX_00308) -->
      <SDG GID="REALIZES">
        <SDX-REF DEST="STRUCTURED-REQ" BASE="SAFEX">ECU_TSR_03</SDX-REF
          >
      </SDG>
    </SDGS>
    </ADMIN-DATA>
   [...]
```

Listing 6.2: AUTOSAR XML representation of the various trace relations

```
<!-- A safety requirement that is decomposed-->
<STRUCTURED-REQ>
  <SHORT-NAME>ECU_TSR_01</SHORT-NAME>
  <LONG-NAME>
    <L-4 L="EN">Ensure CAN Msg received</L-4>
  </LONG-NAME>
  <CATEGORY>SAFETY_TECHNICAL</CATEGORY>
  <ADMIN-DATA>
    <SDGS>
      <SDG GID="SAFEX">
        <SD GID="ASIL">B</SD>
        <SD GID="STATUS">PROPOSED</SD>
      </SDG>
    </SDGS>
  </ADMIN-DATA>
  <TRACE-REFS>
    <TRACE-REF DEST="STRUCTURED-REQ" BASE="SAFEX">SysSafReq05</TRACE-REF>
    <TRACE-REF DEST="STRUCTURED-REQ" BASE="SAFEX">SysSafReq03</TRACE-REF>
    <TRACE-REF DEST="STRUCTURED-REQ" BASE="SAFEX">SysSafReq47</TRACE-REF>
    <!-- optional links for traceability-->
  </TRACE-REFS>
  <TYPE>Valid</TYPE>
  <DESCRIPTION>
    <P>
```

```
      <L-1 L="EN">The CAN message CAN BUS CAN_CL15 shall be received
         correctly.</L-1>
    </P>
  </DESCRIPTION>
  <RATIONALE />
  <DEPENDENCIES />
  <USE-CASE />
  <SUPPORTING-MATERIAL>
    <P>
      <L-1 L="EN">Example for TPS_SAFEX_00302, constr_6202, and
         TPS_SAFEX_00303</L-1>
    </P>
  </SUPPORTING-MATERIAL>
</STRUCTURED-REQ>
<!-- First decomposed technical safety requirement -->
<STRUCTURED-REQ>
  <SHORT-NAME>ECU_TSR_03</SHORT-NAME>
  <LONG-NAME>
    <L-4 L="EN">Ensure correct CAN Bus Msg transformation</L-4>
  </LONG-NAME>
  <CATEGORY>SAFETY_TECHNICAL</CATEGORY>
  <ADMIN-DATA>
    <SDGS>
      <SDG GID="SAFEX">
        <SD GID="ASIL">QM(B)</SD>
        <SD GID="STATUS">PROPOSED</SD>
      </SDG>
      <SDG GID="DECOMPOSITION">
        <SDX-REF DEST="STRUCTURED-REQ" BASE="SAFEX">ECU_TSR_01</SDX-REF>
      </SDG>
      <SDG GID="INDEPENDENCE">
        <SDX-REF DEST="STRUCTURED-REQ" BASE="SAFEX">ECU_TSR_047</SDX-REF>
      </SDG>
      <SDG GID="ALLOCATION">
        <SDX-REF DEST="APPLICATION-SW-COMPONENT-TYPE" BASE="FLM_pkg">/
           FLM_pkg/FLM_swc/FLM</SDX-REF>
      </SDG>
    </SDGS>
  </ADMIN-DATA>
  <TYPE>Valid</TYPE>
  <DESCRIPTION>
    <P>
      <L-1 L="EN">The correct transformation of CAN BUS CAN_CL15 to the
         logical CL15_01 message shall be ensured.</L-1>
    </P>
  </DESCRIPTION>
  <RATIONALE />
  <DEPENDENCIES />
  <USE-CASE />
  <SUPPORTING-MATERIAL>
    <P>
      <L-1 L="EN">Example for TPS_SAFEX_00306]</L-1>
    </P>
  </SUPPORTING-MATERIAL>
</STRUCTURED-REQ>
<!-- Second decomposed technical safety requirement -->
```

```
<STRUCTURED-REQ>
  <SHORT-NAME>ECU_TSR_05</SHORT-NAME>
  <LONG-NAME>
    <L-4 L="EN">CL15_ON failure checks</L-4>
  </LONG-NAME>
  <CATEGORY>SAFETY_TECHNICAL</CATEGORY>
  <ADMIN-DATA>
    <SDGS>
      <SDG GID="SAFEX">
        <SD GID="ASIL">B(B)</SD>
        <SD GID="STATUS">PROPOSED</SD>
      </SDG>
      <SDG GID="DECOMPOSITION">
        <SDX-REF DEST="STRUCTURED-REQ" BASE="SAFEX">ECU_TSR_01</SDX-REF>
      </SDG>
      <SDG GID="INDEPENDENCE">
        <SDX-REF DEST="STRUCTURED-REQ" BASE="SAFEX">ECU_TSR_047</SDX-REF>
      </SDG>
      <SDG GID="MAPS_TO">
        <SDX-REF DEST="TRACEABLE" BASE="SAFEX">SM_E2E</SDX-REF>
      </SDG>
    </SDGS>
  </ADMIN-DATA>
  <TYPE>Valid</TYPE>
  <DESCRIPTION>
    <P>
      <L-1 L="EN">The ECU shall detect any potential communication faults
         affecting the signal CL15ON that could lead to a violation ot
         the safety goal.</L-1>
    </P>
  </DESCRIPTION>
  <RATIONALE />
  <DEPENDENCIES />
  <USE-CASE />
  <SUPPORTING-MATERIAL>
    <P>
      <L-1 L="EN">Example for TPS_SAFEX_00305</L-1>
    </P>
  </SUPPORTING-MATERIAL>
</STRUCTURED-REQ>
<!-- Technical safety requirement that expresses independence -->
<STRUCTURED-REQ>
  <SHORT-NAME>ECU_TSR_047</SHORT-NAME>
  <LONG-NAME>
    <L-4 L="EN">Freedom from interference in signal processing</L-4>
  </LONG-NAME>
  <CATEGORY>SAFETY_TECHNICAL</CATEGORY>
  <ADMIN-DATA>
    <SDGS>
      <SDG GID="SAFEX">
        <SD GID="ASIL">B</SD>
        <SD GID="STATUS">PROPOSED</SD>
      </SDG>
    </SDGS>
  </ADMIN-DATA>
  <TRACE-REFS>
```

```
  <TRACE-REF DEST="STRUCTURED-REQ" BASE="SAFEX">ECU_TSR_03</TRACE-REF>
  <TRACE-REF DEST="STRUCTURED-REQ" BASE="SAFEX">ECU_TSR_05</TRACE-REF>
  <TRACE-REF DEST="STRUCTURED-REQ" BASE="SAFEX">ECU_TSR_01</TRACE-REF>
  <!-- optional links for traceability-->
</TRACE-REFS>
<TYPE>Valid</TYPE>
<DESCRIPTION>
  <P>
    <L-1 L="EN">Independence of signal transformation and communication
       fault detection must be ensured</L-1>
  </P>
</DESCRIPTION>
<RATIONALE />
<DEPENDENCIES />
<USE-CASE />
<SUPPORTING-MATERIAL>
  <P>
    <L-1 L="EN">This safety requirement is part of example for
       TPS_SAFEX_00303</L-1>
  </P>
</SUPPORTING-MATERIAL>
```

7 Safety Measures

Safety of a system is achieved by means of *safety measures* that are applied at various stages of the development process and *safety mechanisms* which are implemented in a number of technologies into the system. This specification considers safety measures beyond the scope of the pure AUTOSAR software stack for a number of reasons:

- Software safety often relies on (external) hardware mechanisms for achieving its safety integrity, such as memory protection and partitioning, ECC/EDC, lock-step modes, external watchdogs, etc. During implementation these context dependencies should be explicitly part of the ”runtime contract” for any software and not just implicitly communicated.
- Error detection and error handling will typically involve a complex interaction between both SW and HW, from the monitoring to the interrupt and handling routine(s), down to the shutoff paths of the actuator. Therefore any software safety mechanism shall be aware of the technological environment, safe state at system level, potential failures and constraints implied by the HW.
- Software integration requires verification of the effectiveness of the safety mechanisms. If the software specification states which safety mechanisms are implemented or which measures are conducted, consistency checks and (semi-) automatic verification becomes possible, which in turn reduces systematic failures.
- Finally, any software is influenced by the HW/platform on which it runs. Understanding and avoiding (systematic) failures is only possible if the system level intend for a safety mechanism is documented, accessible and well understood by the implementors.

AUTOSAR provides already a number of safety mechanisms and features that can be used to implement safe software, for example end to end protection, program flow monitoring, watchdog manager, and so on (see [4] for an overview). These features can be used as target for a mapping of [TPS_SAFEX_00305]. Note that this specification does not specify any constraints on the textual descriptions except the requirements in this section.

[TPS_SAFEX_00401] Definition of Safety Measure or Safety Mechanism ⌈ A safety measure (or safety mechanism) shall be described as `TraceableText`. The `category` attribute shall mark the text block with `SAFETY_MEASURE` or `SAFETY_MECHANISM` respectively. ⌋*(RS_SAFEX_00013, RS_SAFEX_00015, RS_SAFEX_00016, RS_SAFEX_00023)*

[TPS_SAFEX_00402] Unique identifier for safety measures ⌈ A safety measure/mechanism shall receive a unique identifier as `shortName`. The ID shall be unique across the extent of an AUTOSAR project. ⌋*(RS_SAFEX_00017)*

Listing 7.1: Example for the AUTOSAR XML representation of an ASIL attribute at an element

```
<!-- Example safety mechanism -->
```

```
<TRACE>
  <SHORT-NAME>SM_E2E</SHORT-NAME>
  <LONG-NAME>
    <L-4 L="EN">End to End protection of the signal CL15ON</L-4>
  </LONG-NAME>
  <CATEGORY>SAFETY_MECHANISM</CATEGORY>
  <ADMIN-DATA>
    <SDGS>
      <SDG GID="ALLOCATION">
        <SDX-REF DEST="END-TO-END-PROTECTION-SET" BASE="FLM_swc">/FLM_swc/
           FLM/MyEnd2EndProfile</SDX-REF>
      </SDG>
    </SDGS>
  </ADMIN-DATA>
  <P>
    <L-1 L="EN">E2E communication protection enabling the sender to protect
        data and the
    receiver to detect errors and handle them at runtime</L-1>
  </P>
</TRACE>
[...]
```

8 Application Notes

No specific application notes for the current version of this specification.

A Mentioned Class Tables

For the sake of completeness, this chapter contains a set of class tables representing meta-classes mentioned in the context of this document, but which are not contained directly in the scope of describing specific meta-model semantics.

Class	**AdminData**			
Package	M2::MSR::AsamHdo::AdminData			
Note	AdminData represents the ability to express administrative information for an element. This administration information is to be treated as meta-data such as revision id or state of the file. There are basically four kinds of meta-data • The language and/or used languages. • Revision information covering e.g. revision number, state, release date, changes. Note that this information can be given in general as well as related to a particular company. • Document meta-data specific for a company			
Base	ARObject			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
docRevision (ordered)	DocRevision	*	aggr	This allows to denote information about the current revision of the object. Note that information about previous revisions can also be logged here. The entries shall be sorted descendant by date in order to reflect the history. Therefore the most recent entry representing the current version is denoted first. **Tags:** xml.roleElement=true; xml.roleWrapperElement=true; xml.sequenceOffset=50; xml.typeElement=false; xml.typeWrapperElement=false
language	LEnum	0..1	attr	This attribute specifies the master language of the document or the document fragment. The master language is the one in which the document is maintained and from which the other languages are derived from. In particular in case of inconsistencies, the information in the master language is priority. **Tags:** xml.sequenceOffset=20
sdg	Sdg	*	aggr	This property allows to keep special data which is not represented by the standard model. It can be utilized to keep e.g. tool specific data. **Tags:** xml.roleElement=true; xml.roleWrapperElement=true; xml.sequenceOffset=60; xml.typeElement=false; xml.typeWrapperElement=false

Attribute	*Type*	*Mul.*	*Kind*	*Note*
usedLanguages	MultiLanguagePlainText	0..1	aggr	This property specifies the languages which are provided in the document. Therefore it should only be specified in the top level admin data. For each language provided in the document there is one entry in MultilanguagePlainText. The content of each entry can be used for illustration of the language. The used language itself depends on the language attribute in the entry. **Tags:** xml.sequenceOffset=30

Table A.1: AdminData

Class	**Identifiable (abstract)**			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::Identifiable			
Note	Instances of this class can be referred to by their identifier (within the namespace borders). In addition to this, Identifiables are objects which contribute significantly to the overall structure of an AUTOSAR description. In particular, Identifiables might contain Identifiables.			
Base	ARObject, MultilanguageReferrable, Referrable			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
desc	MultiLanguageOverviewParagraph	0..1	aggr	This represents a general but brief (one paragraph) description what the object in question is about. It is only one paragraph! Desc is intended to be collected into overview tables. This property helps a human reader to identify the object in question. More elaborate documentation, (in particular how the object is built or used) should go to "introduction". **Tags:** xml.sequenceOffset=-60
category	CategoryString	0..1	attr	The category is a keyword that specializes the semantics of the Identifiable. It affects the expected existence of attributes and the applicability of constraints. **Tags:** xml.sequenceOffset=-50
adminData	AdminData	0..1	aggr	This represents the administrative data for the identifiable object. **Tags:** xml.sequenceOffset=-40
annotation	Annotation	*	aggr	Possibility to provide additional notes while defining a model element (e.g. the ECU Configuration Parameter Values). These are not intended as documentation but are mere design notes. **Tags:** xml.sequenceOffset=-25

Attribute	*Type*	*Mul.*	*Kind*	*Note*
introduction	DocumentationBlock	0..1	aggr	This represents more information about how the object in question is built or is used. Therefore it is a DocumentationBlock. **Tags:** xml.sequenceOffset=-30
uuid	String	0..1	attr	The purpose of this attribute is to provide a globally unique identifier for an instance of a meta-class. The values of this attribute should be globally unique strings prefixed by the type of identifier. For example, to include a DCE UUID as defined by The Open Group, the UUID would be preceded by "DCE:". The values of this attribute may be used to support merging of different AUTOSAR models. The form of the UUID (Universally Unique Identifier) is taken from a standard defined by the Open Group (was Open Software Foundation). This standard is widely used, including by Microsoft for COM (GUIDs) and by many companies for DCE, which is based on CORBA. The method for generating these 128-bit IDs is published in the standard and the effectiveness and uniqueness of the IDs is not in practice disputed. If the id namespace is omitted, DCE is assumed. An example is "DCE:2fac1234-31f8-11b4-a222-08002b34c003". The uuid attribute has no semantic meaning for an AUTOSAR model and there is no requirement for AUTOSAR tools to manage the timestamp. **Tags:** xml.attribute=true

Table A.2: Identifiable

Class	**Referrable (abstract)**			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::Identifiable			
Note	Instances of this class can be referred to by their identifier (while adhering to namespace borders).			
Base	ARObject			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
shortName	Identifier	1	attr	This specifies an identifying shortName for the object. It needs to be unique within its context and is intended for humans but even more for technical reference. **Tags:** xml.enforceMinMultiplicity=true; xml.sequenceOffset=-100
shortNameFragment	ShortNameFragment	*	aggr	This specifies how the Referrable.shortName is composed of several shortNameFragments. **Tags:** xml.sequenceOffset=-90

Table A.3: Referrable

Class	**Sd**			
Package	M2::MSR::AsamHdo::SpecialData			
Note	This class represents a primitive element in a special data group.			
Base	ARObject			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
gid	NameToken	1	attr	This attributes specifies an identifier. Gid comes from the SGML/XML-Term "Generic Identifier" which is the element name in XML. The role of this attribute is the same as the name of an XML - element. **Tags:** xml.attribute=true
value	VerbatimStringPlain	1	attr	This is the value of the special data. **Tags:** xml.roleElement=false; xml.roleWrapperElement=false; xml.typeElement=false; xml.typeWrapperElement=false
xmlSpace	XmlSpaceEnum	0..1	attr	This attribute is used to signal an intention that in that element, white space should be preserved by applications. It is defined according to xml:space as declared by W3C. **Tags:** xml.attribute=true; xml.attributeRef=true; xml.enforceMinMultiplicity=true; xml.name=space; xml.nsPrefix=xml

Table A.4: Sd

Class	**Sdg**			
Package	M2::MSR::AsamHdo::SpecialData			
Note	Sdg (SpecialDataGroup) is a generic model which can be used to keep arbitrary information which is not explicitly modeled in the meta-model. Sdg can have various contents as defined by sdgContentsType. Special Data should only be used moderately since all elements should be defined in the meta-model. Thereby SDG should be considered as a temporary solution when no explicit model is available. If an sdgCaption is available, it is possible to establish a reference to the sdg structure.			
Base	ARObject			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
gid	NameToken	1	attr	This attributes specifies an identifier. Gid comes from the SGML/XML-Term "Generic Identifier" which is the element name in XML. The role of this attribute is the same as the name of an XML - element. **Tags:** xml.attribute=true

Attribute	Type	Mul.	Kind	Note
sdgCaption	SdgCaption	0..1	aggr	This aggregation allows to assign the properties of Identifiable to the sdg. By this, a shortName etc. can be assigned to the Sdg. **Tags:** xml.sequenceOffset=20
sdgCaptionRef	SdgCaption	0..1	ref	This association allows to reuse an already existing caption. **Tags:** xml.name=SDG-CAPTION-REF; xml.sequenceOffset=25
sdgContentsType	SdgContents	0..1	aggr	This is the content of the Sdg. **Tags:** xml.roleElement=false; xml.roleWrapperElement=false; xml.sequenceOffset=30; xml.typeElement=false; xml.typeWrapperElement=false

Table A.5: Sdg

Class	≪atpMixed≫ SdgContents			
Package	M2::MSR::AsamHdo::SpecialData			
Note	This meta-class represents the possible contents of a special data group. It can be an arbitrary mix of references, of primitive special data and nested special data groups.			
Base	ARObject			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
sd	Sd	0..1	aggr	This is one particular special data element. **Tags:** xml.sequenceOffset=40
sdf	Sdf	0..1	aggr	This is one particular special data element. **Tags:** xml.sequenceOffset=60
sdg	Sdg	0..1	aggr	This aggregation allows to express nested special data groups. By this, any structure can be represented in SpeicalData. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=postBuild xml.sequenceOffset=50
sdx	Referrable	0..1	ref	Reference to any identifiable element. This allows to use Sdg even to establish arbitrary relationships.
sdxf	Referrable	0..1	ref	Additional reference with variant support. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=postBuild

Table A.6: SdgContents

Class	**StructuredReq**			
Package	M2::MSR::Documentation::BlockElements::RequirementsTracing			
Note	This represents a structured requirement. This is intended for a case where specific requirements for features are collected. Note that this can be rendered as a labeled list.			
Base	ARObject, DocumentViewSelectable, Identifiable, MultilanguageReferrable, Paginateable, Referrable, Traceable			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
appliesTo	standardNameEnum	*	attr	This attribute represents the platform the requirement is assigned to. **Tags:** xml.namePlural=APPLIES-TO-DEPENDENCIES; xml.sequenceOffset=25
conflicts	Documentation Block	0..1	aggr	This represents an informal specification of conflicts. **Tags:** xml.sequenceOffset=40
date	DateTime	1	attr	This represents the date when the requirement was initiated. **Tags:** xml.sequenceOffset=5
dependencies	Documentation Block	0..1	aggr	This represents an informal specifiaction of dependencies. Note that upstream tracing should be formalized in the property trace provided by the superclass Traceable. **Tags:** xml.sequenceOffset=30
description	Documentation Block	0..1	aggr	Ths represents the general description of the requirement. **Tags:** xml.sequenceOffset=10
importance	String	1	attr	This allows to represent the importance of the requirement. **Tags:** xml.sequenceOffset=8
issuedBy	String	1	attr	This represents the person, organization or authority which issued the requirement. **Tags:** xml.sequenceOffset=6
rationale	Documentation Block	0..1	aggr	This represents the rationale of the requirement. **Tags:** xml.sequenceOffset=20
remark	Documentation Block	0..1	aggr	This represents an informal remark. Note that this is not modeled as annotation, since these remark is still essential part of the requirement. **Tags:** xml.sequenceOffset=60
supporting Material	Documentation Block	0..1	aggr	This represents an informal specifiaction of the supporting material. **Tags:** xml.sequenceOffset=50

Attribute	Type	Mul.	Kind	Note
testedItem	Traceable	*	ref	This assocation represents the ability to trace on the same specification level. This supports for example the of acceptance tests. **Tags:** xml.sequenceOffset=70
type	String	1	attr	This attribute allows to denote the type of requirement to denote for example is it an "enhancement", "new feature" etc. **Tags:** xml.sequenceOffset=7
useCase	Documentation Block	0..1	aggr	This describes the relevant use cases. Note that formal references to use cases should be done in the trace relation. **Tags:** xml.sequenceOffset=35

Table A.7: StructuredReq

Class	TraceReferrable (abstract)			
Package	M2::MSR::Documentation::BlockElements::RequirementsTracing			
Note	This meta class is intended to add the category to the subclasses of Traceable. Even if the model seems to be a bit awkward, it ensures backwards compatibility of the schema. This approach allows to have subclasses of Traceable which are either Identifiable or only Referrable while still maintaining the consistent sequence of shortName, longName, category.			
Base	ARObject, MultilanguageReferrable, Referrable			
Attribute	Type	Mul.	Kind	Note
–	–	–	–	–

Table A.8: TraceReferrable

Class	Traceable (abstract)			
Package	M2::MSR::Documentation::BlockElements::RequirementsTracing			
Note	This meta class represents the ability to be subject to tracing within an AUTOSAR model. Note that it is expected that its subclasses inherit either from MultilanguageReferrable or from Identifiable. Nevertheless it also inherits from MultilanguageReferrable in order to provide a common reference target for all Traceables.			
Base	ARObject, MultilanguageReferrable, Referrable			
Attribute	Type	Mul.	Kind	Note

Attribute	Type	Mul.	Kind	Note
trace	Traceable	*	ref	This assocation represents the ability to trace to upstream requirements / constraints. This supports for example the bottom up tracing ProjectObjectives <- MainRequirements <- Features <- RequirementSpecs <- BSW/AI **Tags:** xml.sequenceOffset=20

Table A.9: Traceable

Class	TraceableText			
Package	M2::MSR::Documentation::BlockElements::RequirementsTracing			
Note	This meta-class represents the ability to denote a traceable text item such as requirements etc. The following approach appliles: • **shortName** represents the tag for tracing • **longName** represents the head line • **category** represents the kind of the tagged text			
Base	ARObject, DocumentViewSelectable, Identifiable, MultilanguageReferrable, Paginateable, Referrable, Traceable			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
text	Documentation Block	1	aggr	This represents the text to which the tag applies. **Tags:** xml.roleElement=false; xml.roleWrapper Element=false; xml.sequenceOffset=30; xml.type Element=false; xml.typeWrapperElement=false

Table A.10: TraceableText

Class	Xfile			
Package	M2::MSR::Documentation::TextModel::InlineTextElements			
Note	This represents to reference an external file within a documentation.			
Base	ARObject, Referrable, SingleLanguageReferrable			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
tool	String	0..1	attr	This element describes the tool which was used to generate the corresponding Xfile . Kept as a string since no specific syntax can be provided to denote a tool. **Tags:** xml.sequenceOffset=50
toolVersion	String	0..1	attr	This element describes the tool version which was used to generate the corresponding xfile. Kept as a string, since no specific syntax can be specified. **Tags:** xml.sequenceOffset=60

Attribute	*Type*	*Mul.*	*Kind*	*Note*
url	Url	0..1	aggr	This represents the URL of the external file. **Tags:** xml.sequenceOffset=30

Table A.11: Xfile