Document Title	ARXML Serialization Rules
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	779

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

Document Change History			
Date	**Release**	**Changed by**	**Description**
2021-11-25	R21-11	AUTOSAR Release Management	• no content changes
2020-11-30	R20-11	AUTOSAR Release Management	• no content changes
2019-11-28	R19-11	AUTOSAR Release Management	• no content changes • Changed Document Status from Final to published
2018-10-31	4.4.0	AUTOSAR Release Management	• editorial changes
2017-12-08	4.3.1	AUTOSAR Release Management	• update of pattern for AUTOSAR XML Schema location hint
2016-11-30	4.3.0	AUTOSAR Release Management	• Initial document structure

Disclaimer

Table of Contents

References

[1] AUTOSAR XML Schema Production Rules
AUTOSAR_TPS_XMLSchemaProductionRules

[2] Software Component Template
AUTOSAR_TPS_SoftwareComponentTemplate

[3] System Template
AUTOSAR_TPS_SystemTemplate

[4] Specification of ECU Configuration
AUTOSAR_TPS_ECUConfiguration

[5] Meta Model
AUTOSAR_MMOD_MetaModel

[6] Meta Model-generated XML Schema
AUTOSAR_MMOD_XMLSchema

[7] Standardization Template
AUTOSAR_TPS_StandardizationTemplate

[8] Extensible Markup Language (XML), v1.0
http://www.w3.org/TR/REC-xml/

[9] XML Schema 1.0
http://www.w3.org/TR/xmlschema-1

[10] Generic Structure Template
AUTOSAR_TPS_GenericStructureTemplate

[11] Unified Modeling Language: Superstructure, Version 2.0, OMG Available Specification, ptc/05-07-04
http://www.omg.org/cgi-bin/apps/doc?formal/05-07-04

[12] Software Process Engineering Meta-Model Specification
http://www.omg.org/spec/SPEM/2.0/

1 Introduction

This document specifies rules on how AUTOSAR models are serialized into AUTOSAR XML descriptions. The intention of this specification is to support the interoperability between AUTOSAR tools by specifying additional constraints on the AUTOSAR XML descriptions that go beyond the definition of the XML structure that is defined by the AUTOSAR XML schema. Benefits include:

- Comparison of AUTOSAR XML descriptions is simplified by defining a normalized representation that avoids meaningless differences such as indention, character encoding.
- Effort for tool implementation is reduced by restricting the amount of different flavors of XML. E.g. different namespace prefixes, character encoding, files names, etc.

AUTOSAR template specifications define the AUTOSAR Data Exchange Format. Figure 1.1 shows the relationship between the AUTOSAR ARXML Serialization Rules (this specification) and other template specifications:

- The AUTOSAR XML Schema Production Rules [1] and this document focus on the physical representation and the XML data format.
- The Software Component Template [2], System Template [3], ECU Configuration Template [4], etc. address the data structure and its semantics.

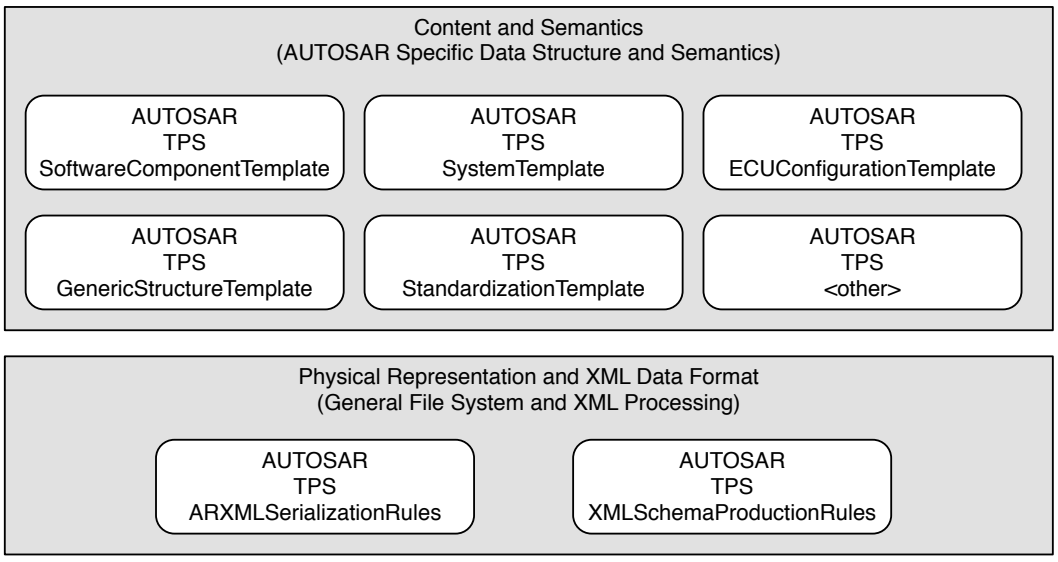

Figure 1.1: Overview Template Specifications

AUTOSAR formalizes and maintains the data structure and semantics of the AUTOSAR Data Exchange Format in the AUTOSAR Meta Model [5]. The mapping between that meta model and the AUTOSAR XML Schema [6] is described in AUTOSAR XML Schema Production Rules [1] (see figure 1.2). An AUTOSAR Tool that produces an AUTOSAR XML Description has to serialize the AUTOSAR model in a way that it

validates successfully against the AUTOSAR XML Schema. Additional constraints that go beyond XML Schema validation are described in this document.

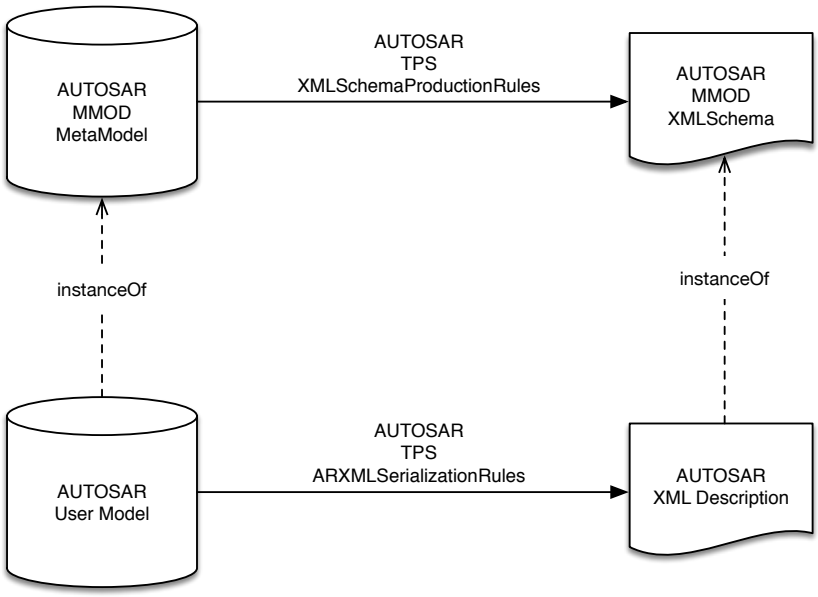

Figure 1.2: Relationship between XML Schema Production Rules and ARXML Serialization Rules

1.1 Document Conventions

Technical terms are typeset in mono spaced font, e.g. `PortPrototype`. As a general rule, plural forms of technical terms are created by adding "s" to the singular form, e.g. `PortPrototype`s. By this means the document resembles terminology used in the AUTOSAR XML Schema.

This document contains constraints in textual form that are distinguished from the rest of the text by a unique numerical constraint ID, a headline, and the actual constraint text starting after the ⌈ character and terminated by the ⌋ character.

The purpose of these constraints is to literally constrain the interpretation of the AUTOSAR meta-model such that it is possible to detect violations of the standardized behavior implemented in an instance of the meta-model (i.e. on M1 level).

Makers of AUTOSAR tools are encouraged to add the numerical ID of a constraint that corresponds to an M1 modeling issue as part of the diagnostic message issued by the tool.

The attributes of the classes introduced in this document are listed in form of class tables. They have the form shown in the example of the top-level element AUTOSAR:

Please note that constraints are not supposed to be enforceable at any given time in an AUTOSAR workflow. During the development of a model, constraints may legitimately be violated because an incomplete model will obviously show inconsistencies.

However, at specific points in the workflow, constraints shall be enforced as a safeguard against misconfiguration.

The points in the workflow where constraints shall be enforced, sometimes also known as the "binding time" of the constraint, are different for each model category, e.g. on the classic platform, the constraints defined for software-components are typically enforced prior to the generation of the RTE while the constraints against the definition of an Ecu extract shall be applied when the Ecu configuration for the Com stack is created.

For each document, possible binding times of constraints are defined and the binding times are typically mentioned in the constraint themselves to give a proper orientation for implementers of AUTOSAR authoring tools.

Let `AUTOSAR` be an example of a typical class table. The first rows in the table have the following meaning:

Class: The name of the class as defined in the UML model.

Package: The UML package the class is defined in. This is only listed to help locating the class in the overall meta model.

Note: The comment the modeler gave for the class (class note). Stereotypes and UML tags of the class are also denoted here.

Base Classes: If applicable, the list of direct base classes.

The headers in the table have the following meaning:

Attribute: The name of an attribute of the class. Note that AUTOSAR does not distinguish between class attributes and owned association ends.

Type: The type of an attribute of the class.

Mul.: The assigned multiplicity of the attribute, i.e. how many instances of the given data type are associated with the attribute.

Kind: Specifies, whether the attribute is aggregated in the class (`aggr` aggregation), an UML attribute in the class (`attr` primitive attribute), or just referenced by it (`ref` reference). Instance references are also indicated (`iref` instance reference) in this field.

Note: The comment the modeler gave for the class attribute (role note). Stereotypes and UML tags of the class are also denoted here.

Please note that the chapters that start with a letter instead of a numerical value represent the appendix of the document. The purpose of the appendix is to support the explanation of certain aspects of the document and does not represent binding conventions of the standard.

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 ([7]).

The representation of requirements in AUTOSAR documents follows the table specified in [TPS_STDT_00078], see Standardization Template, chapter Support for Traceability ([7]).

2 ARXML Serialization Rules

2.1 Physical Level

2.1.1 File separation

[TPS_ASR_00001] File separation ⌈An AUTOSAR model may be shipped in several AUTOSAR XML description files.⌋*()*

Example 2.1

Some files could contain data types others could contain interfaces, etc.

2.1.2 File names

[TPS_ASR_00002] File Name Extension: .arxml ⌈AUTOSAR XML descriptions shall use the file extension ".arxml" (short for AUTOSAR XML).⌋*()*

[TPS_ASR_00003] File Name Length ⌈The maximum length of the filename is restricted to 255 characters.⌋*()*

2.2 Data Format

In order to support a direct comparison of AUTOSAR XML descriptions with a text comparison tool it is essential that the XML is generated in a reliable and standardized manner.

2.2.1 XML Character Encoding

[TPS_ASR_00004] UTF-8 Character Encoding ⌈The character encoding of AUTOSAR XML descriptions shall be UTF-8. No other encodings are allowed.⌋*()*

[TPS_ASR_00005] UTF-8 Encoding in XML Declaration ⌈AUTOSAR XML descriptions shall start with an XML declaration that declares UTF-8 encoding.⌋*()*

Example 2.2

```
<?xml .... encoding="UTF-8"?>
```

[TPS_ASR_00006] Avoid UTF BOM ⌈AUTOSAR XML descriptions should NOT start with a "UTF Byte Order Mask" (BOM).⌋*()*

The byte order mask is a unicode character that can be used at the start of a text stream in order to communicate information about:

- The fact that the stream is encoded in unicode
- Which unicode encoding is used (UTF-8, UTF-16, ...)
- The endianness of the unicode encoding

According to [TPS_ASR_00004] and [TPS_ASR_00005] the character encoding of AUTOSAR XML descriptions shall be UTF-8 and this information shall be explicitly described in the XML declaration. Additionally, UTF-8 doesn't support different endiannesses.

Thus, using a BOM does not add additional information.

2.2.2 XML Version

[TPS_ASR_00007] XML Version 1.0 ⌈AUTOSAR XML descriptions shall conform to XML version 1.0 [8]. No other XML version is allowed.⌋*()*

[TPS_ASR_00008] XML Version 1.0 in XML Declaration ⌈AUTOSAR XML descriptions shall start with an XML declaration that declares XML version 1.0 [8].⌋*()*

Example 2.3

```
<?xml version="1.0" .... ?>
```

2.2.3 XML Comments and Processing Instructions

[TPS_ASR_00009] XML Comments ⌈AUTOSAR XML descriptions may contain XML comments.⌋*()*

Note: XML comments do not contribute to the actual AUTOSAR model. AUTOSAR tools may silently ignore XML comments and do not need to serialize them again.

[TPS_ASR_00010] XML Processing Instructions ⌈An AUTOSAR XML description may contain XML processing instructions. [1]⌋*()*

Note: AUTOSAR tools may silently ignore XML Processing instructions and do not need to serialize them again.

[1] The only exception from this rule is the declaration of the XML version and the XML character encoding. These processing instructions shall be supported as required by [TPS_ASR_00005] and [TPS_ASR_00008]

2.2.4 XML Root Element

Traditionally, AUTOSAR has implemented a three-element version scheme consisting of major, minor, and patch version. Versions specified this way have been used in ARXML files as part of the definition of the `xsi:schemaLocation`, for example:

```
xsi:schemaLocation="http://autosar.org/schema/r4.0 AUTOSAR_4-3-0.xsd"
```

With the advent of the AUTOSAR adaptive platform, AUTOSAR decided to implement a different versioning scheme for the releases of the adaptive platform (the classic platform would just keep the existing approach to versioning).

This new version scheme for the adaptive platform consists of just two elements, the year and month of release.

The original approach was to simply use the two-element scheme of the adaptive releases also for the definition of the `xsi:schemaLocation` for ARXML files containing models for the adaptive platform.

```
xsi:schemaLocation="http://autosar.org/schema/r4.0 AUTOSAR_2017-03.xsd"
```

Over time, this approach would have created a hard-to-disentangle history of three-element and two-element values for `xsi:schemaLocation` and it would have been hard to guess which releases of the AUTOSAR XML Schema were actually backwards-compatible to a given ARXML file.

In order to mitigate the problem, AUTOSAR also decided to invent a completely new versioning scheme for the schema releases, independent of whether the individual schema release would be triggered by the AUTOSAR classic platform or the AUTOSAR adaptive platform.

The new versioning scheme for being used in the `xsi:schemaLocation` foresees the existence of just one element, a positive number that is increased with every AUTOSAR release, whether the release focuses on the classic or adaptive platform does not matter.

```
xsi:schemaLocation="http://autosar.org/schema/r4.0 AUTOSAR_00044.xsd"
```

Each value of the one element in the `xsi:schemaLocation` can be unambiguously identified with a specific AUTOSAR release. Plus, it is still easily possible to understand the backwards compatibility status of a given ARXML file.

The XML schema contains the latest releases of the AUTOSAR standards. This means that there is no dedicated AUTOSAR XML schema that contains only model elements of AUTOSAR classic or adaptive platform. See also figure 2.1.

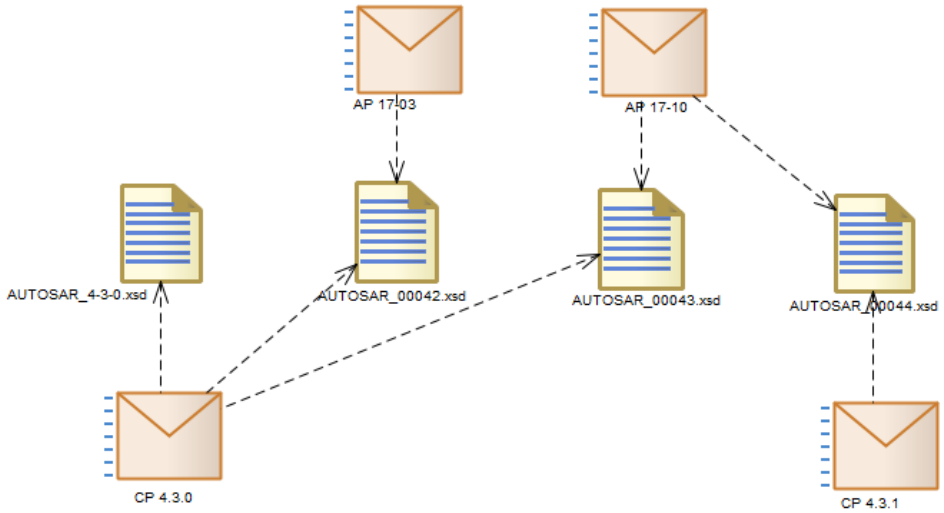

Figure 2.1: Releases of AUTOSAR standards that are contained in the AUTOSAR XML Schema

[TPS_ASR_00011] AUTOSAR XML Namespace ⌈The AUTOSAR XML namespace for all AUTOSAR XML elements and attributes is http://autosar.org/schema/r<major>.<minor>.

The namespace is kept across multiple releases of the AUTOSAR XML Schema as long as backward compatibility is kept. <major> and <minor> are the major and minor version numbers of the AUTOSAR release that starts a sequence of backwards compatible AUTOSAR XML Schema.⌋*()*

Example 2.4

The XML namespace http://autosar.org/schema/r4.0 corresponds to the AUTOSAR XML Schema of AUTOSAR releases 4.0.1. The AUTOSAR XML Schema of the following releases (4.0.2, 4.0.3, 4.1.0, 4.1.1, 4.1.2, 4.1.3, 4,2.1, 4.2.2, 4.3.0, etc.) are intended to be backwards compatible to this release.

[TPS_ASR_00017] AUTOSAR XML Namespace Declaration ⌈The AUTOSAR XML namespace is the default namespace. No namespace prefix shall be applied for AUTOSAR elements.⌋*()*

Example 2.5

<AUTOSAR xmlns="http://autosar.org/schema/r4.0" ... >
instead of
<AR:AUTOSAR xmlns:AR="http://autosar.org/schema/r4.0" ... >

[TPS_ASR_00018] No Third-Party XML Namespaces ⌈The only valid XML namespaces that are allowed in AUTOSAR XML descriptions are:

- the AUTOSAR XML namespace (http://autosar.org/schema/r<major>.<minor>) [TPS_ASR_00017] and
- the XML Schema Instance namespace (http://www.w3.org/2001/XMLSchema-instance)

No other Third-Party XML namespaces are allowed.⌋*()*

[TPS_ASR_00012] AUTOSAR Revision Declaration ⌈The AUTOSAR XML description shall declare the AUTOSAR revision which was the basis for its **creation** via the schema location hint URI [TPS_ASR_00013] that is mapped to the AUTOSAR namespace [TPS_ASR_00011] in the `xsi:schemaLocation` attribute. The attribute `xsi:schemaLocation` and the declaration of the AUTOSAR schema location hint for the AUTOSAR namespace is mandatory.⌋*()*

Note: According to the W3C XML Schema specification [9], chapter 4.3.2 "How schema definitions are located on the Web", the attribute xsi:schemalocation specifies pairs of URI references (one for the XML namespace, and one for a hint as to the location of a schema document defining names for that XML namespace). It is expected that a tool that validates a AUTOSAR XML descriptions is able to identify an appropriate XML Schema document in its own resources.

This approach allows the validation of AUTOSAR XML descriptions against newer AUTOSAR XML Schema as long as the AUTOSAR XML Schema is backwards compatible. Additionally, the tool can try to validate the AUTOSAR XML description against an older AUTOSAR XML Schema as long as the AUTOSAR XML description does not use newer features.

Example 2.6

Example of a AUTOSAR revision declaration for AUTOSAR revision 4.3.0:

```
<AUTOSAR ...
 xsi:schemaLocation="http://autosar.org/schema/r4.0 AUTOSAR_4-3-0.xsd>
</AUTOSAR>
```

[TPS_ASR_00013] Pattern for AUTOSAR XML Schema location hint URI ⌈The `AUTOSAR XML Schema location hint URI` in the AUTOSAR XML description shall be the file name of a XML Schema document provided by AUTOSAR. This file name follows the pattern:

```
AUTOSAR_{number}.xsd
```

{number} corresponds to the specific AUTOSAR release the given AUTOSAR XML Schema belongs to.

In particular no path shall be part of the `AUTOSAR XML Schema location hint URI`.⌋*()*

Example of AUTOSAR XML Root Element is provided in listing 2.1:

Listing 2.1: AUTOSAR XML Root Element

```
<?xml version="1.0" encoding="UTF8"?>
<AUTOSAR
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="{AUTOSAR␣XML␣Namespace}␣{Revision␣Hint␣URI}"
    xmlns="{AUTOSAR␣XML␣Namespace}">
        ...
</AUTOSAR>
```

2.2.5 XML Formating / Indention

The formating and indention that is specified in this section does not change the semantics of the AUTOSAR model. The main purpose is the reduction of meaningless differences when comparing two AUTOSAR XML descriptions using textual diff tools.

[TPS_ASR_00019] Formating of AUTOSAR XML Descriptions ⌈The XML description should be formatted as shown in Table 2.1⌋*()*

Applied to	Strategy	Description
default approach	*NewLine*: Element is a block of its own	NewLine means in particular: • indentation should be 2 characters per level • the start tag of the element should be on a new line • the XML attributes should be sorted alphabetically. If more than one XML attribute, each one should be on its own line • the start should be indented according to the nesting level of XML tag • the end tag should be on a new line and indented like the start tag • the content should be indented one step more than the start tag
Primitives (either modeled as UML-attribute or as aggregation of a primitive	*OneLine* Element is displayed in one line	The element should start on a new line. The end tag should be in the same line as the start tag and the content of the element.
Properties of ≪atpMixedString≫	*InLine*: Element is floating within text	Surrounding whitespace of the element should not be changed. No new line should be inserted before or after the tags. Whitespace within the element should not be changed. In the following example the element <E> is formatted according to the *InLine* approach. <L-1 L="EN">This is <E>bold</E> style </L-1>
VerbatimString elements with xml:space set to preserve	*keepWhitespace*	White space in the element should be kept as is.

▽

△

elements with no `xml:space` or set to `default`	*normalizeWhitespace*	Normalize whitespace includes: • leading and trailing whitespace should be removed • consecutive white spaces should be replaced by a single blank • no wrapping should be performed • carriage returns should be replaced by blank • child(inline)-elements should be treated as one non whitespace character

Table 2.1: Approaches for formating XML serialization

The following example 2.2 illustrates these approaches:

Listing 2.2: Serialization Example

```
<UNIT>
  <SHORT-NAME>Perc</SHORT-NAME>                          <!-- OneLine -->
  <DESC>                                                 <!-- NewLine -->
    <L-2 L="EN">a percentage...</L-2>                    <!-- OneLine -->
  </DESC>
  <DISPLAY-NAME>%</DISPLAY-NAME>                         <!-- OneLine -->
</UNIT>
<UNIT>
  <SHORT-NAME>PercPerSec</SHORT-NAME>                    <!-- OneLine -->
  <DESC>                                                 <!-- NewLine -->
    <L-2 L="EN">time-derivative of percent</L-2>         <!-- NewLine
                                                              NormalizeWhitespace


                                                         -->
  </DESC>
  <DISPLAY-NAME>%/s</DISPLAY-NAME>                       <!-- OneLine -->
</UNIT>
```

[TPS_ASR_00015] Empty elements represented by start-end tag pairs ⌈Empty elements should be serialized as start/end tag, not as 'emptytag'.⌋*()*

Example 2.7

An empty VALUE tag should be serialized as <VALUE></VALUE> instead of the technically possible alternative <VALUE/>.

[TPS_ASR_00016] No empty wrappers ⌈Some attributes and references in AUTOSAR models are mapped to a hierarchy of two or more XML elements. The AUTOSAR XML description should not contain incomplete hierarchies. The semantics of those incomplete hierarchies is equivalent to "the value is not set".

These rules applies for attributes, aggregations and references for which the following XML Schema production rules apply [1]:

- [TPS_XMLSPR_00008] XML Schema production rule: composite property representation (1111)

- [TPS_XMLSPR_00009] XML Schema production rule: composite property representation (1101)
- [TPS_XMLSPR_00023] XML Schema production rule: composite property representation (1100)
- [TPS_XMLSPR_00022] XML Schema production rule: composite property representation (1011)
- [TPS_XMLSPR_00010] XML Schema production rule: composite property representation (1001)
- [TPS_XMLSPR_00011] XML Schema production rule: composite property representation (0111)
- [TPS_XMLSPR_00012] XML Schema production rule: composite property representation (0101)
- [TPS_XMLSPR_00014] XML Schema production rule: composite property representation (0011)
- [TPS_XMLSPR_00017] XML Schema production rule: reference property representation with role wrapper element

⌋()

Example of a **valid** AUTOSAR XML description according to [TPS_ASR_00016]:

Listing 2.3: Valid example for hierarchy

```
<?xml version="1.0" encoding="UTF-8"?>
<AUTOSAR
xmlns="http://autosar.org/schema/r4.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://autosar.org/schema/r4.0␣AUTOSAR_00044.xsd">
    <!-- .... -->
</AUTOSAR>
```

Example of an **invalid** AUTOSAR XML description according to [TPS_ASR_00016]:

Listing 2.4: Invalid example for hierarchy

```
<?xml version="1.0" encoding="UTF-8"?>
<AUTOSAR
xmlns="http://autosar.org/schema/r4.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://autosar.org/schema/r4.0␣AUTOSAR_00044.xsd">
    <AR-PACKAGES>
    </AR-PACKAGES>
</AUTOSAR>
```

The AUTOSAR meta model explicitly defines if the order of elements that are owned by an attribute is relevant. The order of an attribute is relevant if

- the attribute is owned by a mixed content class (a class with stereotype «atpMixed» or «atpMixedString» as defined by [TPS_GST_00024], [TPS_GST_00025], [TPS_GST_00032] in [10]) or
- the attribute with upper multiplicity > 1 is flagged as {ordered} according to the UML specification [11].

Tools shall not change the order of elements whose order is semantically relevant. However, if the order of elements is not relevant, then a tool may serialize the elements in an arbitrary order. This often results in meaningless differences when comparing AUTOSAR XML descriptions using textual diff tools. In order to reduce those meaningless differences the following rules should apply.

[TPS_ASR_00014] Sorting elements if their order is semantically meaningless ⌈Attributes with upper multiplicity > 1 whose order of elements is semantically meaningless (not flagged as {ordered} and not owned by a class with stereotype «atpMixed» or «atpMixedString») should be serialized using the following heuristics:

1. If the AUTOSAR meta model defines an atp.Splitkey (see [TPS_GST_00050] in [10]) at the aggregation then
   the contained elements shall be sorted alphabetically in ascending order using a key that is calculated according to the expression mentioned in the atp.Splitkey.
   E.g. if atp.Splitkey="shortName,variationPoint.shortLabel" then the elements are sorted by a key that is calculated by concatenation of the values of the OCL expressions in the atp.Splitkey:
   shortName + "," + variationPoint.shortLabel.
2. If no atp.Splitkey is defined, then the following expression for calculation of the key is assumed: shortName,shortLabel,variationPoint.shortLabel. If the shortName, shortLabel or variationPoint.shortLabel is not defined then its value is assumed to be an empty string.

If the attribute is of kind reference then the following rule applies

1. The absolute short name path of the referenced target shall be used even if it is a relative reference. See also [TPS_GST_00169] and [TPS_GST_00352] in [10].

The strategy for calculation of the sort key might not be able to calculate unique keys for all sets of elements. This is a known limitation. For those cases the producing tool should define its own custom strategy in order to ensure deterministic serialization of elements for which the order is semantically meaningless.⌋ *()*

3 Glossary

Artifact This is a Work Product Definition that provides a description and definition for tangible work product types. Artifacts may be composed of other artifacts ([12]).

At a high level, an artifact is represented as a single conceptual file.

AUTOSAR Tool This is a software tool which supports one or more tasks defined as AUTOSAR tasks in the methodology. Depending on the supported tasks, an AUTOSAR tool can act as an authoring tool, a converter tool, a processor tool or as a combination of those (see separate definitions).

AUTOSAR Authoring Tool An AUTOSAR Tool used to create and modify AUTOSAR XML Descriptions. Example: System Description Editor.

AUTOSAR Converter Tool An AUTOSAR Tool used to create AUTOSAR XML files by converting information from other AUTOSAR XML files. Example: ECU Flattener

AUTOSAR Definition This is the definition of parameters which can have values. One could say that the parameter values are Instances of the definitions. But in the meta model hierarchy of AUTOSAR, definitions are also instances of the meta model and therefore considered as a description. Examples for AUTOSAR definitions are: `EcucParameterDef`, `PostBuildVariantCriterion`, `SwSystemconst`.

AUTOSAR XML Description In AUTOSAR this means "filled Template". In fact an AUTOSAR XML description is the XML representation of an AUTOSAR model.

The AUTOSAR XML description can consist of several files. Each individual file represents an AUTOSAR partial model and shall validate successfully against the AUTOSAR XML schema.

AUTOSAR Meta-Model This is an UML2.0 model that defines the language for describing AUTOSAR systems. The AUTOSAR meta-model is an UML representation of the AUTOSAR templates. UML2.0 class diagrams are used to describe the attributes and their interrelationships. Stereotypes, UML tags and OCL expressions (object constraint language) are used for defining specific semantics and constraints.

AUTOSAR Meta-Model Tool The AUTOSAR Meta-Model Tool is the tool that generates different views (class tables, list of constraints, diagrams, XML Schema etc.) on the AUTOSAR meta-model.

AUTOSAR Model This is a representation of an AUTOSAR product. The AUTOSAR model represents aspects suitable to the intended use according to the AUTOSAR methodology.

Strictly speaking, this is an instance of the AUTOSAR meta-model. The information contained in the AUTOSAR model can be anything that is representable according to the AUTOSAR meta-model.

AUTOSAR Partial Model In AUTOSAR, the possible partitioning of models is marked in the meta-model by ≪atpSplitable≫. One partial model is represented in an AUTOSAR XML description by one file. The partial model does not need to fulfill all semantic constraints applicable to an AUTOSAR model.

AUTOSAR Processor Tool An AUTOSAR Tool used to create non-AUTOSAR files by processing information from AUTOSAR XML files. Example: RTE Generator

AUTOSAR Specification Element An AUTOSAR Specification Element is a named element that is part of an AUTOSAR specification. Examples: requirement, constraint, specification item, class or attribute in the meta model, methodology, deliverable, methodology activity, model element, bsw module etc.

AUTOSAR Template The term "Template" is used in AUTOSAR to describe the format different kinds of descriptions. The term template comes from the idea, that AUTOSAR defines a kind of form which shall be filled out in order to describe a model. The filled form is then called the description.

In fact the AUTOSAR templates are now defined as a meta-model.

AUTOSAR Validation Tool A specialized AUTOSAR Tool which is able to check an AUTOSAR model against the rules defined by a profile.

AUTOSAR XML Schema This is a W3C XML schema that defines the language for exchanging AUTOSAR models. This Schema is derived from the AUTOSAR meta-model. The AUTOSAR XML Schema defines the AUTOSAR data exchange format.

Blueprint This is a model from which other models can be derived by copy and refinement. Note that in contrast to meta model resp. types, this process is *not* an instantiation.

Instance Generally this is a particular exemplar of a model or of a type.

Life Cycle Life Cycle is the course of development/evolutionary stages of a model element during its life time.

Meta-Model This defines the building blocks of a model. In that sense, a Meta-Model represents the language for building models.

Meta-Data This includes pertinent information about data, including information about the authorship, versioning, access-rights, timestamps etc.

Model A Model is an simplified representation of reality. The model represents the aspects suitable for an intended purpose.

Partial Model This is a part of a model which is intended to be persisted in one particular artifact.

Pattern in GST This is an approach to simplify the definition of the meta model by applying a model transformation. This transformation creates an enhanced model out of an annotated model.

Profile Authoring Support Data Data that is used for efficient authoring of a profile. E.g. list of referable constraints, meta-classes, meta-attributes or other reusable model assets (blueprints)

Profile Authoring Tool A specialized `AUTOSAR Tool` which focuses on the authoring of profiles for data exchange points. It e.g. provides support for the creation of profiles from scratch, modification of existing profiles or composition of existing profiles.

Profile Compatibility Checker Tool A specialized `AUTOSAR Tool` which focuses on checking the compatibility of profiles for data exchange. Note that this compatibility check includes manual compatibility checks by engineers and automated assistance using more formal algorithms.

Profile Consistency Checker Tool A specialized `AUTOSAR Tool` which focuses on checking the consistency of profiles.

Property A property is a structural feature of an object. As an example a “connector” has the properties “receive port” and “send port”

Properties are made variant by the ≪`atpVariation`≫.

Prototype This is the implementation of a role of a type within the definition of another type. In other words a type may contain Prototypes that in turn are typed by "Types". Each one of these prototypes becomes an instance when this type is instantiated.

Type A type provides features that can appear in various roles of this type.

Value This is a particular value assigned to a “Definition”.

Variability Variability of a system is its quality to describe a set of variants. These variants are characterized by variant specific property settings and / or selections. As an example, such a system property selection manifests itself in a particular “receive port” for a connection.

This is implemented using the ≪`atpVariation`≫.

Variant A system variant is a concrete realization of a system, so that all its properties have been set respectively selected. The software system has no variability anymore with respect to the binding time.

This is implemented using `EvaluatedVariantSet`.

Variation Binding A variant is the result of a variation binding process that resolves the variability of the system by assigning particular values/selections to all the system’s properties.

This is implemented by `VariationPoint`.

Variation Binding Time The variation binding time determines the step in the methodology at which the variability given by a set of variable properties is resolved.

This is implemented by `vh.LatestBindingtime` at the related properties.

Variation Definition Time The variation definition time determines the step in the methodology at which the variation points are defined.

Variation Point A variation point indicates that a property is subject to variation. Furthermore, it is associated with a condition and a binding time which define the system context for the selection / setting of a concrete variant.

This is implemented by `VariationPoint`.

A Change History

A.1 Change History of R4.3.0

A.1.1 Added Traceables

ID	Heading	Origin in R4.2.2
[TPS_ASR_00001]	File separation	Extends: [TR_IOAT_00010] AUTOSAR tool SHALL support sets of files
[TPS_ASR_00002]	File Name Extension: .arxml	Subset of: [TR_IOAT_00062] Authoring tool SHALL support well defined serialization (incl. explanatory text)
[TPS_ASR_00003]	File Name Length	Subset of: [TR_IOAT_00069]
[TPS_ASR_00004]	UTF-8 Character Encoding	Replaces: [TR_APRXML_00049] UTF-8 Character Encoding
[TPS_ASR_00005]	UTF-8 Encoding in XML Declaration	Replaces: [TR_APRXML_00050] UTF-8 Encoding in XML Declaration
[TPS_ASR_00006]	Avoid UTF BOM	Replaces: [TR_APRXML_00051] Avoid UTF BOM
[TPS_ASR_00007]	XML version 1.0	Subset of: [TR_IOAT_00012] AUTOSAR tool SHALL support AUTOSAR XML descriptions
[TPS_ASR_00008]	XML version 1.0 in XML Declaration	Subset of: [TR_IOAT_00012] AUTOSAR tool SHALL support AUTOSAR XML descriptions
[TPS_ASR_00009]	XML Comments	Subset of: [TR_IOAT_00062] Authoring tool SHALL support well defined serialization (incl. explanatory text)
[TPS_ASR_00010]	XML Processing Instructions	Subset of: [TR_IOAT_00062] Authoring tool SHALL support well defined serialization (incl. explanatory text)
[TPS_ASR_00011]	AUTOSAR XML Namespace	Supplements: [TR_APRXML_00035] XML schema version, Subset of: [TR_APRXML_00052] AUTOSAR namespace declaration

▽

△

[TPS_ASR_00012]	AUTOSAR Revision Declaration	Subset of: [TR_IOAT_00062] Authoring tool SHALL support well defined serialization (incl. explanatory text)
[TPS_ASR_00013]	Pattern for AUTOSAR Revision Hint URI	Subset of: [TR_IOAT_00062] Authoring tool SHALL support well defined serialization (incl. explanatory text)
[TPS_ASR_00014]	Order of Elements	Subset of: [TR_IOAT_00062] Authoring tool SHALL support well defined serialization (incl. explanatory text)
[TPS_ASR_00015]	Empty elements represented by start-end tag pairs	Subset of: [TR_IOAT_00062] Authoring tool SHALL support well defined serialization (incl. explanatory text)
[TPS_ASR_00016]	No empty wrappers	Replaces: [TR_IOAT_00075] No empty wrappers
[TPS_ASR_00017]	No empty wrappers	Replaces: [TR_IOAT_00075] No empty wrappers
[TPS_ASR_00018]	No Third-Party XML Namespaces	Subset of: [1] chapter “XML description production”
[TPS_ASR_00019]	Formating of AUTOSAR XML Descriptions	Subset of: [TR_IOAT_00062] Authoring tool SHALL support well defined serialization (incl. explanatory text)

Table A.1: Changed Traceables in 4.3.0

A.1.2 Changed Traceables

none

A.1.3 Deleted Traceables

none

A.2 Change History of R4.3.1

A.2.1 Added Traceables in R4.3.1

none

A.2.2 Changed Traceables in R4.3.1

none

A.2.3 Deleted Traceables in R4.3.1

none

A.2.4 Added Constraints in R4.3.1

none

A.2.5 Changed Constraints in R4.3.1

none

A.2.6 Deleted Constraints in R4.3.1

none

A.3 Change History of R4.4.0

A.3.1 Added Traceables in R4.4.0

none

A.3.2 Changed Traceables in R4.4.0

none

A.3.3 Deleted Traceables in R4.4.0

none

A.3.4 Added Constraints in R4.4.0

none

A.3.5 Changed Constraints in R4.4.0

none

A.3.6 Deleted Constraints in R4.4.0

none

A.4 Change History of R19-11

A.4.1 Added Traceables in R19-11

none

A.4.2 Changed Traceables in R19-11

none

A.4.3 Deleted Traceables in R19-11

none

A.4.4 Added Constraints in R19-11

none

A.4.5 Changed Constraints in R19-11

none

A.4.6 Deleted Constraints in R19-11

none

A.5 Change History of R20-11

A.5.1 Added Traceables in R20-11

none

A.5.2 Changed Traceables in R20-11

none

A.5.3 Deleted Traceables in R20-11

none

A.5.4 Added Constraints in R20-11

none

A.5.5 Changed Constraints in R20-11

none

A.5.6 Deleted Constraints in R20-11

none

A.6 Change History of R21-11

A.6.1 Added Traceables in R21-11

none

A.6.2 Changed Traceables in R21-11

none

A.6.3 Deleted Traceables in R21-11

none

A.6.4 Added Constraints in R21-11

none

A.6.5 Changed Constraints in R21-11

none

A.6.6 Deleted Constraints in R21-11

none

B 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	**AUTOSAR**			
Package	M2::AUTOSARTemplates::AutosarTopLevelStructure			
Note	Root element of an AUTOSAR description, also the root element in corresponding XML documents. **Tags:**xml.globalElement=true			
Base	*ARObject*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
adminData	AdminData	0..1	aggr	This represents the administrative data of an Autosar file. **Tags:**xml.sequenceOffset=10

▽

△

Class	**AUTOSAR**			
arPackage	ARPackage	*	aggr	This is the top level package in an AUTOSAR model. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=arPackage.shortName, arPackage.variationPoint.shortLabel vh.latestBindingTime=blueprintDerivationTime xml.sequenceOffset=30
fileInfoComment	FileInfoComment	0..1	aggr	This represents a possibility to provide a structured comment in an AUTOSAR file. **Stereotypes:** atpStructuredComment **Tags:** xml.roleElement=true xml.sequenceOffset=-10 xml.typeElement=false
introduction	DocumentationBlock	0..1	aggr	This represents an introduction on the Autosar file. It is intended for example to represent disclaimers and legal notes. **Tags:**xml.sequenceOffset=20

Table B.1: AUTOSAR

Class	***PortPrototype*** (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	Base class for the ports of an AUTOSAR software component. The aggregation of PortPrototypes is subject to variability with the purpose to support the conditional existence of ports.			
Base	*ARObject*, *AtpBlueprintable*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	*AbstractProvidedPortPrototype*, *AbstractRequiredPortPrototype*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
logAndTraceMessageCollectionSet	LogAndTraceMessageCollectionSet	0..1	ref	Reference to a collection of Log or Trace messages that will be used by the application. **Tags:**atp.Status=draft

Table B.2: PortPrototype

Primitive	**VerbatimString**			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::PrimitiveTypes			
Note	This primitive represents a string in which white-space needs to be preserved. **Tags:** xml.xsd.customType=VERBATIM-STRING xml.xsd.type=string xml.xsd.whiteSpace=preserve			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
blueprintValue	String	0..1	attr	This represents a description that documents how the value shall be defined when deriving objects from the blueprint. **Tags:** atp.Status=draft xml.attribute=true

▽

△

Primitive	**VerbatimString**			
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.name=space xml.nsPrefix=xml

Table B.3: VerbatimString