Document Title	Log And Trace Extract Template
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	1024

Document Status	published
Part of AUTOSAR Standard	Foundation
Part of Standard Release	R22-11

Document Change History			
Date	**Release**	**Changed by**	**Description**
2022-11-24	R22-11	AUTOSAR Release Management	• Added modeling support for Privacy Flags
2021-11-25	R21-11	AUTOSAR Release Management	• Initial Release

Disclaimer

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

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

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

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

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

Contents

References

[1] Log and Trace Protocol Specification
AUTOSAR_PRS_LogAndTraceProtocol

[2] System Template
AUTOSAR_TPS_SystemTemplate

[3] Specification of Manifest
AUTOSAR_TPS_ManifestSpecification

[4] Software Component Template
AUTOSAR_TPS_SoftwareComponentTemplate

[5] Generic Structure Template
AUTOSAR_TPS_GenericStructureTemplate

1 Introduction

The Log and Trace functionality (DLT) in AUTOSAR allows Classic Platform SW-Cs/Modules, Adaptive Platform Applications/Functional Clusters to forward log and trace information onto external Dlt Log Viewers if supported on the underlying platform.

The format of the DLT messages is described in more detail in the Log and Trace Protocol Specification [1].

Please note that the configuration of the source of log and trace messages (application or module that produces the logging information) is described in chapter 2.

The ARXML format of `DltMessage`s is described in chapter 3. This format can be used in Classic Platform System Template [2] descriptions and Adaptive Platform Manifest [3] descriptions to define log and trace messages sent out by Classic Platform SW-Cs/Modules or Adaptive Platform Applications/Functional Clusters.

The description of the `DltMessage` is particularly important to support the Non-Verbose Mode of the Log And Trace Protocol. To reduce the amount of traffic on the bus, no meta data about the interpretation of the `DltMessage` is transmitted on the communications bus in the Non-Verbose Mode. Instead, the ARXML file that describes the `DltMessage` holds the information about the interpretation of the `DltMessage`.

The following figure shows the placement of the Log and Trace Extract in the AUTOSAR Methodology.
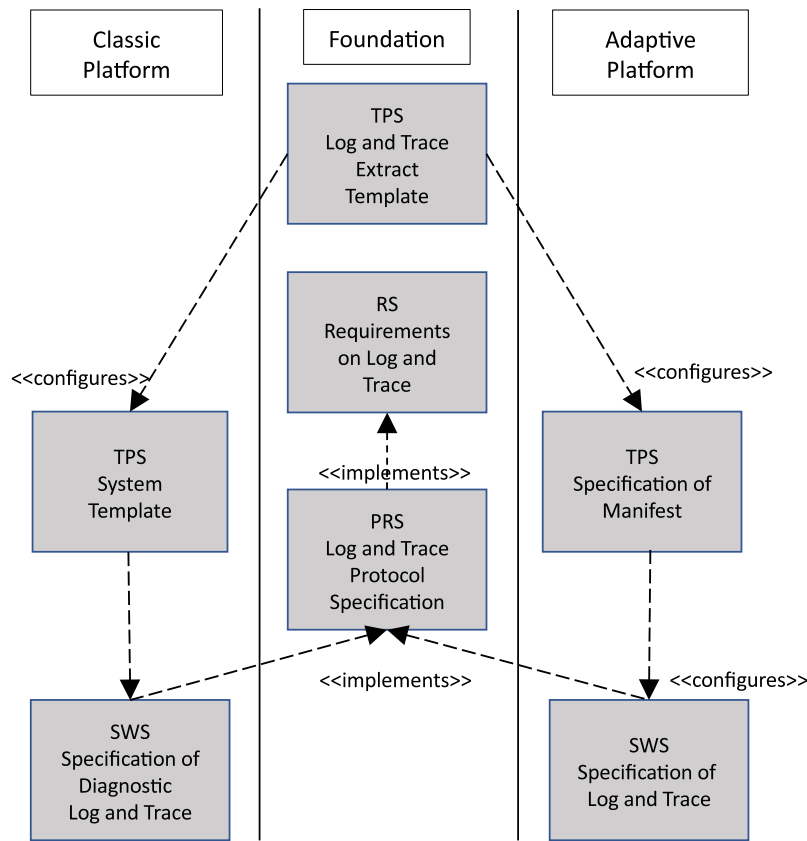
Figure 1.1: Methodological placement of Log and Trace Extract

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.

2 Dlt Ecu

This chapter describes the `DltEcu` that represents the Classic Platform "ECU" or Adaptive Platform "Machine" on which `DltApplication`s are deployed that are generating `DltMessage`s. The `DltApplication` and `DltContext` are used to identify the source of the log and trace message.
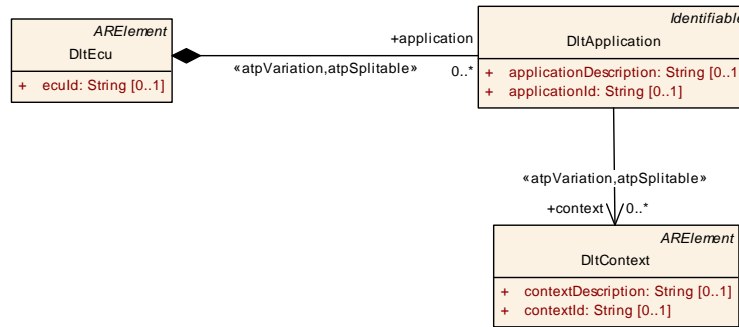

Figure 2.1: Modeling of DltEcu

Class	DltEcu			
Package	M2::AUTOSARTemplates::LogAndTraceExtract			
Note	This element represents an Ecu or Machine that produces logging and tracing information. **Tags:**atp.recommendedPackage=DltEcus			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
application	DltApplication	*	aggr	Application on DltEcu that provides log or trace data. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=application.shortName, application.variation Point.shortLabel vh.latestBindingTime=systemDesignTime
ecuId	String	0..1	attr	This attribute defines the name of the ECU for use within the Dlt protocol.

Table 2.1: DltEcu

Class	DltApplication			
Package	M2::AUTOSARTemplates::LogAndTraceExtract			
Note	This meta-class represents the application from which the log and trace message originates.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	DltEcu.application			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
application Description	String	0..1	attr	This attribute can be used to describe the applicationId that is used in the log and trace message in more detail.
applicationId	String	0..1	attr	This attribute identifies the SW-C/BSW module in the log and trace message.

$\bigtriangledown$

△

Class	DltApplication			
context	DltContext	*	ref	Definition of ContextIds for the Application. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=context.dltContext, context.variationPoint.shortLabel vh.latestBindingTime=systemDesignTime

Table 2.2: DltApplication

Class	DltContext			
Package	M2::AUTOSARTemplates::LogAndTraceExtract			
Note	This meta-class represents the Context that groups Log and Trace Messages that are generated by an application. **Tags:**atp.recommendedPackage=DltContexts			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
contextDescription	String	0..1	attr	This attribute can be used to describe the contextId that is used in the log and trace message in more detail.
contextId	String	0..1	attr	This attribute is used to group log and trace messages produced by an application to distinguish functionality.
dltMessage	DltMessage	*	ref	Group of Log and Trace Messages assigned to the Dlt Context **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=dltMessage.dltMessage, dltMessage.variationPoint.shortLabel vh.latestBindingTime=systemDesignTime

Table 2.3: DltContext

[constr_5294] Existence of `DltEcu.ecuId` ⌈For each `DltEcu`, the attribute `ecuId` shall exist when the Log And Trace Extract is created.⌋*()*

[constr_5295] Existence of `DltApplication.context` ⌈Each `DltApplication` shall reference at least one `DltContext` in the role `context` when the Log And Trace Extract is created.⌋*()*

[constr_5296] Existence of `DltApplication.applicationId` ⌈For each `DltApplication`, the attribute `applicationId` shall exist when the Log And Trace Extract is created.⌋*()*

[constr_5297] Existence of `DltApplication.applicationDescription` ⌈For each `DltApplication`, the attribute `applicationDescription` shall exist when the Log And Trace Extract is created.⌋*()*

[constr_5298] Existence of `DltContext.contextId` ⌈For each `DltContext`, the attribute `contextId` shall exist when the Log And Trace Extract is created.⌋*()*

[constr_5299] Existence of `DltContext.contextDescription` ⌈For each `DltContext`, the attribute `contextDescription` shall exist when the Log And Trace Extract is created.⌋*()*

[constr_5300] Existence of `DltContext.dltMessage` ⌈Each `DltContext` shall reference at least one `DltMessage` in the role `dltMessage` when the Log And Trace Extract is created.⌋*()*

3 Dlt Message

This chapter describes the modeling of `DltMessage`s. Please note that some information of the Dlt message format that is described in [1] like the Message Info and the Number of Arguments can be derived from the `DltMessage` itself. Other parts in the Dlt message format represent the source of the log or trace messages (e.g. `ecuId`, `applicationId`, `contextId`) and this information can be derived from the `DltEcu`, the `DltApplication` that is aggregated by the `DltEcu` and the `DltContext` that is referenced by the `DltApplication`.
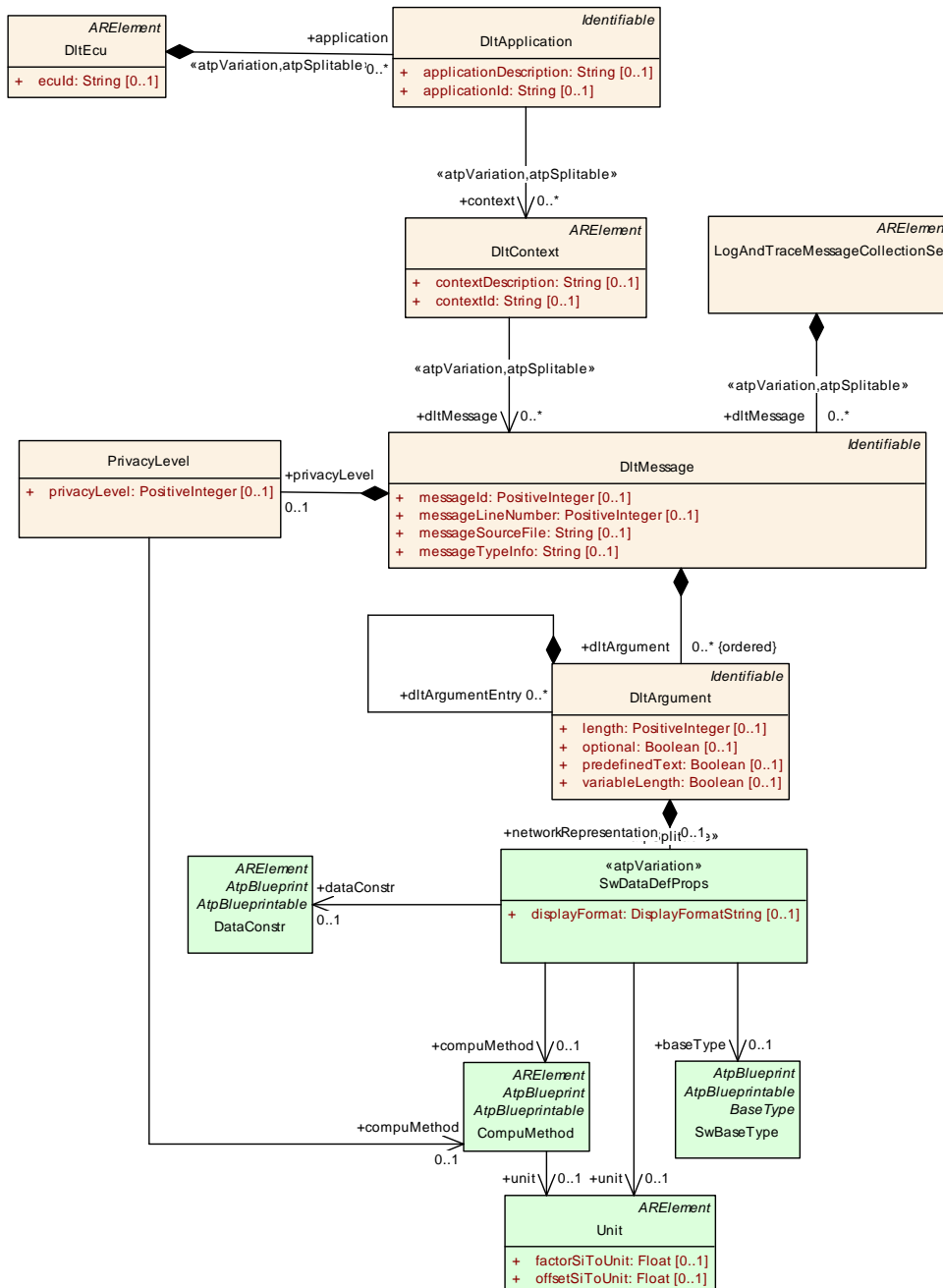
Figure 3.1: Modeling of DltMessages

Class	LogAndTraceMessageCollectionSet			
Package	M2::AUTOSARTemplates::LogAndTraceExtract			
Note	Collection of DltMessages **Tags:** atp.recommendedPackage=LogAndTraceMessageCollectionSets			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
dltMessage	DltMessage	*	aggr	Collection of DltMessages in the DltMessageCollection Set. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=dltMessage.shortName, dlt Message.variationPoint.shortLabel vh.latestBindingTime=systemDesignTime

Table 3.1: LogAndTraceMessageCollectionSet

Class	DltMessage			
Package	M2::AUTOSARTemplates::LogAndTraceExtract			
Note	This element defines a DltMessage.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	LogAndTraceMessageCollectionSet.dltMessage			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
dltArgument (ordered)	DltArgument	*	aggr	Ordered collection of DltArguments in the DltMessage.
messageId	PositiveInteger	0..1	attr	This attribute defines the unique Id for the DltMessage.
messageLine Number	PositiveInteger	0..1	attr	This attribute describes the position in the source file in which this log message was called.
messageSource File	String	0..1	attr	This attribute describes the source file in which this log message was called.
messageType Info	String	0..1	attr	This attribute describes the message Type
privacyLevel	PrivacyLevel	0..1	aggr	The Privacy Level helps to identify the Log and Trace content towards the degree of privacy to it.

Table 3.2: DltMessage

[TPS_DLTXT_00001] Log or trace message representation ⌈One log or trace message is represented by one `DltMessage` element.⌋ *()*

[constr_5301] Existence of `DltMessage.messageId` ⌈For each `DltMessage`, the attribute `messageId` shall exist when the Log And Trace Extract is created.⌋ *()*

[TPS_DLTXT_00002] DLT message ⌈The `DltMessage` is described with the following information:

- Message Type (MSTP) - shall be derived from `DltMessage.messageTypeInfo`

- Message Info (MSIN) - shall be derived from `DltMessage.messageTypeInfo`

- Source file - shall be derived from `DltMessage.messageSourceFile`

- Line Number - shall be derived from `DltMessage.messageLineNumber`

⌋*()*

Please note that the source file and the line number described in [TPS_DLTXT_00002] are not applicable to standardized `DltMessage`s or to `DltMessage`s that are meant to be reused in multiple locations.

Class	DltArgument			
Package	M2::AUTOSARTemplates::LogAndTraceExtract			
Note	This element defines an Argument in a DltMessage.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	DltArgument.dltArgumentEntry, DltMessage.dltArgument			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
dltArgument Entry	DltArgument	*	aggr	This aggregation is used to describe subElements of a Dlt Argument that defines a Structure.
length	PositiveInteger	0..1	attr	Describes the DltArgument length in case of Arrays and Strings in number of BaseTypes.
network Representation	SwDataDefProps	0..1	aggr	Definition of the networkRepresentation of the Dlt Argument. **Stereotypes:** atpSplitable **Tags:**atp.Splitkey=networkRepresentation
optional	Boolean	0..1	attr	This attribute defines whether the argument is optional or not. If set to true, the argument can be omitted from the payload of a DLT message.
predefinedText	Boolean	0..1	attr	This attribute defines whether the DltArgument is a predefinedText (Static Data).
variableLength	Boolean	0..1	attr	This attribute defines whether the length of the Dlt Argument is variable (determined at runtime) or not.

Table 3.3: DltArgument

[TPS_DLTXT_00003] User data of the log or trace message ⌈The user data of the log or trace message shall be represented by `DltArgument`s that are ordered in the `DltMessage`.⌋*()*

[TPS_DLTXT_00004] `DltArgument` kinds ⌈Each `DltArgument` can be either:

- predefined text (static)

- assembled data

⌋*()*

Please note that the Log and Trace Protocol Specification [1] is using the terms "Static-Data" for predefined text and "Non-Static-Data" for assembled data.

[TPS_DLTXT_00005] predefined text `DltArgument` ⌈If a `DltArgument` is a predefined text, then the `DltArgument` shall only contain a `shortName`, a `desc` and the `predefinedText`. The text shall be placed into the `desc` field. The `predefinedText` attribute of the `DltArgument` shall be set to true.⌋*()*

[TPS_DLTXT_00006] assembled data `DltArgument` ⌈If a `DltArgument` represents assembled data then `DltArgument.networkRepresentation.unit` and the `DltArgument.networkRepresentation.baseType` shall be defined.⌋*()*

Please note that only the assembled data of the `DltMessage` is send to external Dlt Log Viewers. The "predefined text" is available only in the ARXML File.

[TPS_DLTXT_00014] Semantics of `DltArgument.optional` ⌈The `optional` attribute defines whether the `DltArgument` is optional (true) or mandatory (false) in the Dlt payload.⌋ *()*

[constr_5302] Restriction in usage of `DltArgument.optional` attribute ⌈The `optional` attribute shall not be set in a `DltArgument` that represents an array dimension.⌋ *()*

The meaning of the term "array dimension" in [constr_5302] refers to table Table 3.4.

[TPS_DLTXT_00009] Semantics of `DltArgument.length` ⌈The `length` attribute defines the length of the `DltArgument` of an Array or a String. If the `DltArgument.variableLength` is set to true this attribute defines the maximum length of the `DltArgument`.⌋ *()*

Please note that for primitive types like Boolean, Signed Integer, Unsigned Integer, and Float the length of the `DltArgument` shall be derived from the `baseTypeSize`.

[constr_5303] Restriction of `baseTypeSize` of a `DltArgument` ⌈The `baseTypeSize` in the `networkRepresentation` of a `DltArgument` is restricted to 8, 16, 32, and 64 Bits.⌋ *()*

[TPS_DLTXT_00010] Semantics of `DltArgument.length` in case of a String ⌈If the `networkRepresentation` of a `DltArgument` is a String which is described with a `SwBaseType` that has a String `baseTypeEncoding` then the `length` attribute specifies the text size in characters. Please note the size in bytes depends on the encoding in the corresponding `SwBaseType`.⌋ *()*

[TPS_DLTXT_00011] Description of `DltArgument` with one-dimensional Array type ⌈A `DltArgument` that has a one-dimensional Array type is described by a `DltArgument` that has the `length` attribute set to a value. The data type of the array is described by the `SwBaseType` in the `networkRepresentation`.⌋ *()*

Please note that a `DltArgument` of type String is described as an Array of characters.

[TPS_DLTXT_00012] Description of `DltArgument` with n-dimensional Array type ⌈A `DltArgument` that has a n-dimensional Array type is described by a `DltArgument` that aggregates further `DltArgument`s in the role `dltArgumentEntry` and all the aggregated `DltArgument`s have the `length` attribute set to a value. Each `dltArgumentEntry` defines one dimension in the Array that is represented by the aggregating `DltArgument`.

The `length` attribute of the aggregating `DltArgument` defines the number of dimensions.

The `length` attribute in the `dltArgumentEntry` defines the number of entries in the dimension.

The data type of the Array is described by the `SwBaseType` in the `networkRepresentation` of the aggregating `DltArgument`.⌋*()*

[constr_5304] Datatype of an Array ⌈The `dltArgumentEntry` that is aggregated by a `DltArgument` that has the `length` attribute set to a value (represents an Array) shall not define a `SwBaseType` in the `networkRepresentation` since the data type of the Array is described by the `SwBaseType` in the `networkRepresentation` of the aggregating `DltArgument`.⌋*()*

[TPS_DLTXT_00013] Description of `DltArgument` with Structure type ⌈A `DltArgument` that has a Structure type is described by a `DltArgument` that aggregates further `DltArgument`s in the role `dltArgumentEntry` and that does not have the `length` attribute set to a value.⌋*()*

The following table summarizes the modeling of the different Payload data types:

Type	length	dltArgumentEntry	SwBaseType of top level DltArgument
Predefined Text	NA	NA	NA
primitive Type	NA	NA	D
String	D	NA	D
1-dimensional Array	D	NA	D
n-dimensional Array	D	D	D
Struct	NA	D	NA

Table 3.4: Allowed usage of attributes for description of payload data types

The following table summarizes the usage of `dltArgumentEntry`:

DltArgumentEntry type	length	dltArgumentEntry	SwBaseType of DltArgumentEntry
Struct member	NA	D	D
Array dimension	D	D	NA

Table 3.5: Allowed usage of attributes in case of a `dltArgumentEntry`

The following settings apply in table 3.4 and table 3.5:

D Attribute can be **defined** in the scope of this element.

NA Attribute is **not applicable** for usage in the scope of this element.

[constr_5098] Allowed `SwDataDefProps` attributes for `DltArgument.networkRepresentation` ⌈

Attributes of `SwDataDefProps`	networkRepresentation
`annotation`	N/A
`baseType`	D
`compuMethod`	D
`dataConstr`	D

▽

$\triangle$

displayFormat	D
displayPresentation	N/A
invalidValue	N/A
swComparisonVariable	N/A
swHostVariable	N/A
swTextProps	N/A
unit	D

⌋*()*

Please note that the set of attributes of `SwDataDefProps` listed in [constr_5098] represents the subset that is visible in the *AUTOSAR foundation*.

The following settings apply in [constr_5098]:

D Attribute can be **defined** in the scope of this element.

NA Attribute is **not applicable** for usage in the scope of this element.

Please note that the `DltMessage` is not necessary transmitted over the network, even if the `DltArgument` defines a `networkRepresentation`. The logging information may be provided onto the communication bus, the console, or to the file system.

The `dataConstr` attribute can be used to describe value ranges of the `DltArgument`. More details about Data Constraints can be found in the Software Component Template [4]. Please note that the Data Constraints are irrelevant for the Dlt Protocol.

The `displayFormat` attribute defines a format specifier for the display of values in tools. More details about the potential value settings can be found in the Generic Structure Template [5].

[constr_5305] CompuMethod in DltArgument.networkRepresentation ⌈The `CompuMethod` that is used in the `networkRepresentation` of a `DltArgument` is limited to `category` TEXTTABLE.⌋*()*

With the TEXTTABLE `CompuMethod` it is possible to define an enumeration as `networkRepresentation` of a `DltArgument`. More details can be found in the Software Component Template [4].

[TPS_DLTXT_00007] Description of a String as networkRepresentation of a DltArgument ⌈If the `networkRepresentation` of a `DltArgument` is a String this can be described with the two `SwBaseType` elements:

- `baseTypeEncoding` (according to [TPS_SWCT_01845] [4])

- `baseTypeSize`

⌋*()*

[TPS_DLTXT_00008] Standardized values of DltMessage.messageTypeInfo ⌈The following values of attributes `DltMessage`.`messageTypeInfo` are standardized by AUTOSAR:

- DLT_LOG_OFF

- DLT_LOG_FATAL

- DLT_LOG_ERROR

- DLT_LOG_WARN

- DLT_LOG_INFO

- DLT_LOG_DEBUG

- DLT_LOG_VERBOSE

- DLT_TRACE_VARIABLE

- DLT_TRACE_FUNCTION_IN

- DLT_TRACE_FUNCTION_OUT

- DLT_TRACE_STATE

- DLT_TRACE_VFB

- DLT_NW_TRACE_IPC

- DLT_NW_TRACE_CAN

- DLT_NW_TRACE_FLEXRAY

- DLT_NW_TRACE_MOST

- DLT_NW_TRACE_ETHERNET

- DLT_NW_TRACE_SOMEIP

- DLT_NW_TRACE_0x7

- DLT_NW_TRACE_0x8

- DLT_NW_TRACE_0x9

- DLT_NW_TRACE_0x10

- DLT_NW_TRACE_0x11

- DLT_NW_TRACE_0x12

- DLT_NW_TRACE_0x13

- DLT_NW_TRACE_0x14

- DLT_NW_TRACE_0x15

- DLT_CONTROL_REQUEST

- DLT_CONTROL_RESPONSE

⌋()

Please note that only the values defined in [TPS_DLTXT_00008] are allowed to be used in `DltMessage.messageTypeInfo`. Custom values are currently not allowed.

Class	PrivacyLevel			
Package	M2::AUTOSARTemplates::LogAndTraceExtract			
Note	This meta-class defines the Privacy Level for a Log and Trace content.			
Base	*ARObject*			
Aggregated by	DltMessage.privacyLevel			
Attribute	**Type**	*Mult.*	**Kind**	**Note**
compuMethod	CompuMethod	0..1	ref	Reference to CompuMethod of category TEXTTABLE that defines the supported user-defined privacy levels.
privacyLevel	PositiveInteger	0..1	attr	The value that represents the privacy level and is transported in the Extension Header.

Table 3.6: PrivacyLevel

[constr_5340] Range of `DltMessage.privacyLevel.privacyLevel` ⌈The value of `DltMessage.privacyLevel.privacyLevel` shall be in the range between 0 and 255.⌋ *()*

[constr_5341] Range of `PrivacyLevel.compuMethod` ⌈The `CompuMethod` that is referenced from `PrivacyLevel` in the role `compuMethod` shall have the category TEXTTABLE.⌋ *()*

The following example shows an ARXML file that defines a `LogAndTraceMessageCollectionSet` with a single `DltMessage`. Based on the `messageId`, the receiver can reassemble all predefined data of this Dlt message:

- position in source code (Line = 72 in demo.c)

- predefined text: "Temperature measurement"

- assembled data 1 and unit: measurement_point (no unit)

- assembled data 2 and unit: reading (Kelvin)

- assembled data 3 and unit: 3-dimensional array that describes the temperature distribution in a room (Kelvin)

```xml
<?xml version="1.0" encoding="utf-8"?>
<AUTOSAR xsi:schemaLocation="http://autosar.org/schema/r4.0 AUTOSAR_00050.
   xsd" xmlns="http://autosar.org/schema/r4.0" xmlns:xsi="http://www.w3.org
   /2001/XMLSchema-instance">
  <AR-PACKAGES>
   <AR-PACKAGE>
     <SHORT-NAME>Dlt_Example</SHORT-NAME>
     <ELEMENTS>
       <DLT-ECU>
         <SHORT-NAME>ECU1</SHORT-NAME>
         <APPLICATIONS>
           <DLT-APPLICATION>
             <SHORT-NAME>App1</SHORT-NAME>
             <APPLICATION-DESCRIPTION>Application 1</APPLICATION-
                DESCRIPTION>
             <APPLICATION-ID>APPL</APPLICATION-ID>
             <CONTEXTS>
               <DLT-CONTEXT-REF-CONDITIONAL>
                 <DLT-CONTEXT-REF DEST="DLT-CONTEXT">/Dlt_Example/CTX1</
                    DLT-CONTEXT-REF>
               </DLT-CONTEXT-REF-CONDITIONAL>
             </CONTEXTS>
           </DLT-APPLICATION>
         </APPLICATIONS>
         <ECU-ID>ECU1</ECU-ID>
       </DLT-ECU>
       <DLT-CONTEXT>
         <SHORT-NAME>CTX1</SHORT-NAME>
         <CONTEXT-DESCRIPTION>Context 1</CONTEXT-DESCRIPTION>
         <CONTEXT-ID>CTX1</CONTEXT-ID>
         <DLT-MESSAGES>
           <DLT-MESSAGE-REF-CONDITIONAL>
             <DLT-MESSAGE-REF DEST="DLT-MESSAGE">/Dlt_Example/
                DltMessageCollectionSetExample/DltMessageWithID_1</DLT-
                MESSAGE-REF>
           </DLT-MESSAGE-REF-CONDITIONAL>
         </DLT-MESSAGES>
       </DLT-CONTEXT>
       <LOG-AND-TRACE-MESSAGE-COLLECTION-SET>
         <SHORT-NAME>DltMessageCollectionSetExample</SHORT-NAME>
         <DLT-MESSAGES>
           <DLT-MESSAGE>
             <SHORT-NAME>DltMessageWithID_1</SHORT-NAME>
             <DLT-ARGUMENTS>
               <DLT-ARGUMENT>
                 <SHORT-NAME>Temperature_measurement</SHORT-NAME>
                 <DESC>
                   <L-2 L="EN">Temperature measurement</L-2>
                 </DESC>
                 <PREDEFINED-TEXT>true</PREDEFINED-TEXT>
               </DLT-ARGUMENT>
               <DLT-ARGUMENT>
                 <SHORT-NAME>measurement_point</SHORT-NAME>
                 <DESC>
                   <L-2 L="EN">Temperature measurement</L-2>
                 </DESC>
```

```
<NETWORK-REPRESENTATION>
  <SW-DATA-DEF-PROPS-VARIANTS>
    <SW-DATA-DEF-PROPS-CONDITIONAL>
      <BASE-TYPE-REF DEST="SW-BASE-TYPE">/
          AUTOSAR_Platform/BaseTypes/uint8</BASE-TYPE-REF>
      <UNIT-REF DEST="UNIT">/AUTOSAR/AISpecification/
          Units/NoUnit</UNIT-REF>
    </SW-DATA-DEF-PROPS-CONDITIONAL>
  </SW-DATA-DEF-PROPS-VARIANTS>
</NETWORK-REPRESENTATION>
</DLT-ARGUMENT>
<DLT-ARGUMENT>
  <SHORT-NAME>reading</SHORT-NAME>
  <DESC>
    <L-2 L="EN">reading</L-2>
  </DESC>
  <NETWORK-REPRESENTATION>
    <SW-DATA-DEF-PROPS-VARIANTS>
      <SW-DATA-DEF-PROPS-CONDITIONAL>
        <BASE-TYPE-REF DEST="SW-BASE-TYPE">/
            AUTOSAR_Platform/BaseTypes/float32</BASE-TYPE-
            REF>
        <UNIT-REF DEST="UNIT">/AUTOSAR/AISpecification/
            Units/Kelvin</UNIT-REF>
      </SW-DATA-DEF-PROPS-CONDITIONAL>
    </SW-DATA-DEF-PROPS-VARIANTS>
  </NETWORK-REPRESENTATION>
</DLT-ARGUMENT>
<DLT-ARGUMENT>
  <!-- Example for a n-dimensional array: "temperature␣
      distribution␣in␣a␣room";
          There are a number of temperature probes
            distributed in a room:
          There are 4 layers along the hight of the room (
            z-coordinate);
          There are 5 rows in a layer along the depth of
            the room (y-coordinate);
          There are 6 probes in a row along the width of
            the room (x-coordinate);
          -> 120 probes;
  -->
  <SHORT-NAME>My_3dim_Array_4x5x6</SHORT-NAME>
  <DESC>
    <L-2 L="EN">Heat distribution map of MyRoom</L-2>
  </DESC>
  <DLT-ARGUMENT-ENTRYS>
    <DLT-ARGUMENT>
      <SHORT-NAME>FirstDim_ArrayOfLayers</SHORT-NAME>
      <DESC>
        <L-2 L="EN">HeatMap_z</L-2>
      </DESC>
      <LENGTH>4</LENGTH>
      <!-- The length attribute in the dltArgumentEntry
          defines the number of entries in the dimension. --
          >
    </DLT-ARGUMENT>
```

```xml
            <DLT-ARGUMENT>
              <SHORT-NAME>SecondDim_ArrayOfRows</SHORT-NAME>
              <DESC>
                <L-2 L="EN">HeatMap_y</L-2>
              </DESC>
              <LENGTH>5</LENGTH>
              <!-- The length attribute in the dltArgumentEntry
                  defines the number of entries in the dimension. --
                  >
            </DLT-ARGUMENT>
            <DLT-ARGUMENT>
              <SHORT-NAME>ThirdDim_ArrayOfProbes</SHORT-NAME>
              <DESC>
                <L-2 L="EN">HeatMap_x</L-2>
              </DESC>
              <LENGTH>6</LENGTH>
              <!-- The length attribute in the dltArgumentEntry
                  defines the number of entries in the dimension. --
                  >
            </DLT-ARGUMENT>
          </DLT-ARGUMENT-ENTRYS>
          <LENGTH>3</LENGTH>
          <!-- The length attribute in the DltArgument defines the
              number of Dimensions. -->
          <NETWORK-REPRESENTATION>
            <SW-DATA-DEF-PROPS-VARIANTS>
              <SW-DATA-DEF-PROPS-CONDITIONAL>
                <BASE-TYPE-REF DEST="SW-BASE-TYPE">/
                    AUTOSAR_Platform/BaseTypes/float32</BASE-TYPE-
                    REF>
                <UNIT-REF DEST="UNIT">/AUTOSAR/AISpecification/
                    Units/Kelvin</UNIT-REF>
              </SW-DATA-DEF-PROPS-CONDITIONAL>
            </SW-DATA-DEF-PROPS-VARIANTS>
          </NETWORK-REPRESENTATION>
        </DLT-ARGUMENT>
      </DLT-ARGUMENTS>
      <MESSAGE-ID>1</MESSAGE-ID>
      <MESSAGE-LINE-NUMBER>72</MESSAGE-LINE-NUMBER>
      <MESSAGE-SOURCE-FILE>demo.c</MESSAGE-SOURCE-FILE>
      <MESSAGE-TYPE-INFO>DLT_LOG_DEBUG</MESSAGE-TYPE-INFO>
    </DLT-MESSAGE>
   </DLT-MESSAGES>
  </LOG-AND-TRACE-MESSAGE-COLLECTION-SET>
 </ELEMENTS>
 </AR-PACKAGE>
</AR-PACKAGES>
```

Listing 3.1: Example for the definition of a LogAndTraceMessageCollectionSet

The following example shows an ARXML file in which one `DltArgument` represents a String and a second `DltArgument` represents an Enumeration.

```xml
<?xml version="1.0" encoding="utf-8"?>
```

```xml
<AUTOSAR xsi:schemaLocation="http://autosar.org/schema/r4.0_AUTOSAR_00050.
    xsd" xmlns="http://autosar.org/schema/r4.0" xmlns:xsi="http://www.w3.org
    /2001/XMLSchema-instance">
  <AR-PACKAGES>
    <AR-PACKAGE>
      <SHORT-NAME>TraceMessageExample</SHORT-NAME>
      <ELEMENTS>
        <LOG-AND-TRACE-MESSAGE-COLLECTION-SET>
          <SHORT-NAME>TraceMessages</SHORT-NAME>
          <DLT-MESSAGES>
            <DLT-MESSAGE>
              <SHORT-NAME>ExecutionManagerProcessStateChangeMsg</SHORT-NAME
                >
              <DLT-ARGUMENTS>
                <DLT-ARGUMENT>
                  <SHORT-NAME>ProcessId</SHORT-NAME>
                  <DESC>
                    <L-2 L="EN">Process id as given by the OS</L-2>
                  </DESC>
                  <NETWORK-REPRESENTATION>
                    <SW-DATA-DEF-PROPS-VARIANTS>
                      <SW-DATA-DEF-PROPS-CONDITIONAL>
                        <BASE-TYPE-REF DEST="SW-BASE-TYPE">/
                          AUTOSAR_Platform/BaseTypes/uint32</BASE-TYPE-REF
                          >
                        <UNIT-REF DEST="UNIT">/AUTOSAR/AISpecification/
                          Units/NoUnit</UNIT-REF>
                      </SW-DATA-DEF-PROPS-CONDITIONAL>
                    </SW-DATA-DEF-PROPS-VARIANTS>
                  </NETWORK-REPRESENTATION>
                </DLT-ARGUMENT>
                <DLT-ARGUMENT>
                  <SHORT-NAME>ProcessName</SHORT-NAME>
                  <DESC>
                    <L-2 L="EN">Process name</L-2>
                  </DESC>
                  <LENGTH>32</LENGTH>
                  <NETWORK-REPRESENTATION>
                    <SW-DATA-DEF-PROPS-VARIANTS>
                      <SW-DATA-DEF-PROPS-CONDITIONAL>
                        <BASE-TYPE-REF DEST="SW-BASE-TYPE">/BaseTypes/
                          Utf8BaseType</BASE-TYPE-REF>
                        <UNIT-REF DEST="UNIT">/AUTOSAR/AISpecification/
                          Units/NoUnit</UNIT-REF>
                      </SW-DATA-DEF-PROPS-CONDITIONAL>
                    </SW-DATA-DEF-PROPS-VARIANTS>
                  </NETWORK-REPRESENTATION>
                </DLT-ARGUMENT>
                <DLT-ARGUMENT>
                  <SHORT-NAME>State</SHORT-NAME>
                  <DESC>
                    <L-2 L="EN">Process State Change</L-2>
                  </DESC>
                  <NETWORK-REPRESENTATION>
                    <SW-DATA-DEF-PROPS-VARIANTS>
                        <SW-DATA-DEF-PROPS-CONDITIONAL>
```

```xml
                    <BASE-TYPE-REF DEST="SW-BASE-TYPE">/
                      AUTOSAR_Platform/BaseTypes/uint8</BASE-TYPE-REF>
                    <COMPU-METHOD-REF DEST="COMPU-METHOD">/CompuMethods
                      /ProcessState</COMPU-METHOD-REF>
                    <UNIT-REF DEST="UNIT">/AUTOSAR/AISpecification/
                      Units/NoUnit</UNIT-REF>
                  </SW-DATA-DEF-PROPS-CONDITIONAL>
                </SW-DATA-DEF-PROPS-VARIANTS>
              </NETWORK-REPRESENTATION>
            </DLT-ARGUMENT>
          </DLT-ARGUMENTS>
          <MESSAGE-ID>1</MESSAGE-ID>
          <MESSAGE-TYPE-INFO>DLT_TRACE_STATE</MESSAGE-TYPE-INFO>
        </DLT-MESSAGE>
      </DLT-MESSAGES>
    </LOG-AND-TRACE-MESSAGE-COLLECTION-SET>
  </ELEMENTS>
</AR-PACKAGE>
<AR-PACKAGE>
  <SHORT-NAME>CompuMethods</SHORT-NAME>
  <ELEMENTS>
    <COMPU-METHOD>
      <SHORT-NAME>ProcessState</SHORT-NAME>
      <LONG-NAME>
        <L-4 L="EN">Process state enumeration</L-4>
      </LONG-NAME>
      <CATEGORY>TEXTTABLE</CATEGORY>
      <UNIT-REF DEST="UNIT">NoUnit</UNIT-REF>
      <COMPU-INTERNAL-TO-PHYS>
        <COMPU-SCALES>
          <COMPU-SCALE>
            <DESC>
              <L-2 L="EN">Idle Process state</L-2>
            </DESC>
            <LOWER-LIMIT INTERVAL-TYPE="CLOSED">0</LOWER-LIMIT>
            <UPPER-LIMIT INTERVAL-TYPE="CLOSED">0</UPPER-LIMIT>
            <COMPU-CONST>
              <VT>Idle</VT>
            </COMPU-CONST>
          </COMPU-SCALE>
          <COMPU-SCALE>
            <DESC>
              <L-2 L="EN">Starting Process state</L-2>
            </DESC>
            <LOWER-LIMIT INTERVAL-TYPE="CLOSED">1</LOWER-LIMIT>
            <UPPER-LIMIT INTERVAL-TYPE="CLOSED">1</UPPER-LIMIT>
            <COMPU-CONST>
              <VT>Starting</VT>
            </COMPU-CONST>
          </COMPU-SCALE>
          <COMPU-SCALE>
            <DESC>
              <L-2 L="EN">Running Process state</L-2>
            </DESC>
            <LOWER-LIMIT INTERVAL-TYPE="CLOSED">2</LOWER-LIMIT>
            <UPPER-LIMIT INTERVAL-TYPE="CLOSED">2</UPPER-LIMIT>
```

```
                    <COMPU-CONST>
                       <VT>Running</VT>
                    </COMPU-CONST>
                 </COMPU-SCALE>
                 <COMPU-SCALE>
                    <DESC>
                       <L-2 L="EN">Terminating Process state</L-2>
                    </DESC>
                    <LOWER-LIMIT INTERVAL-TYPE="CLOSED">3</LOWER-LIMIT>
                    <UPPER-LIMIT INTERVAL-TYPE="CLOSED">3</UPPER-LIMIT>
                    <COMPU-CONST>
                       <VT>Terminating</VT>
                    </COMPU-CONST>
                 </COMPU-SCALE>
                 <COMPU-SCALE>
                    <DESC>
                       <L-2 L="EN">Terminated Process state</L-2>
                    </DESC>
                    <LOWER-LIMIT INTERVAL-TYPE="CLOSED">4</LOWER-LIMIT>
                    <UPPER-LIMIT INTERVAL-TYPE="CLOSED">4</UPPER-LIMIT>
                    <COMPU-CONST>
                       <VT>Terminated</VT>
                    </COMPU-CONST>
                 </COMPU-SCALE>
               </COMPU-SCALES>
             </COMPU-INTERNAL-TO-PHYS>
           </COMPU-METHOD>
        </ELEMENTS>
      </AR-PACKAGE>
      <AR-PACKAGE>
        <SHORT-NAME>BaseTypes</SHORT-NAME>
        <ELEMENTS>
          <SW-BASE-TYPE>
            <SHORT-NAME>Utf8BaseType</SHORT-NAME>
            <CATEGORY>FIXED_LENGTH</CATEGORY>
            <BASE-TYPE-SIZE>8</BASE-TYPE-SIZE>
            <BASE-TYPE-ENCODING>UTF-8</BASE-TYPE-ENCODING>
          </SW-BASE-TYPE>
        </ELEMENTS>
      </AR-PACKAGE>
    </AR-PACKAGES>
</AUTOSAR>
```

Listing 3.2: Example for String and Enumeration DltArguments

A   History of Constraints and Specification Items

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

A.1   Constraint and Specification Item History of this document according to AUTOSAR Release R21-11

A.1.1   Added Traceables in R21-11

Number	Heading
[TPS_DLTXT_00001]	Log or trace message representation
[TPS_DLTXT_00002]	DLT message
[TPS_DLTXT_00003]	User data of the log or trace message
[TPS_DLTXT_00004]	`DltArgument` kinds
[TPS_DLTXT_00005]	predefined text `DltArgument`
[TPS_DLTXT_00006]	assembled data `DltArgument`
[TPS_DLTXT_00007]	Description of a String as `networkRepresentation` of a `DltArgument`
[TPS_DLTXT_00008]	Standardized values of `DltMessage.messageTypeInfo`
[TPS_DLTXT_00009]	Semantics of `DltArgument.length`
[TPS_DLTXT_00010]	Semantics of `DltArgument.length` in case of a String
[TPS_DLTXT_00011]	Description of `DltArgument` with one-dimensional Array type
[TPS_DLTXT_00012]	Description of `DltArgument` with n-dimensional Array type
[TPS_DLTXT_00013]	Description of `DltArgument` with Structure type
[TPS_DLTXT_00014]	Semantics of `DltArgument.optional`

Table A.1: Added Traceables in R21-11

A.1.2   Changed Traceables in R21-11

A.1.3   Deleted Traceables in R21-11

A.1.4 Added Constraints in R21-11

Number	Heading
[constr_5098]	Usage of `DltArgument.networkRepresentation`
[constr_5294]	Existence of `DltEcu.ecuId`
[constr_5295]	Existence of `DltApplication.context`
[constr_5296]	Existence of `DltApplication.applicationId`
[constr_5297]	Existence of `DltApplication.applicationDescription`
[constr_5298]	Existence of `DltContext.contextId`
[constr_5299]	Existence of `DltContext.contextDescription`
[constr_5300]	Existence of `DltContext.dltMessage`
[constr_5301]	Existence of `DltMessage.messageId`
[constr_5302]	Restriction in usage of `DltArgument.optional` attribute
[constr_5303]	Restriction of `baseTypeSize` of a `DltArgument`
[constr_5304]	Datatype of an Array
[constr_5305]	`CompuMethod` in `DltArgument.networkRepresentation`

Table A.2: Added Constraints in R21-11

A.1.5 Changed Constraints in R21-11

A.1.6 Deleted Constraints in R21-11

A.2 Constraint and Specification Item History of this document according to AUTOSAR Release R22-11

A.2.1 Added Traceables in R22-11

A.2.2 Changed Traceables in R22-11

A.2.3   Deleted Traceables in R22-11

A.2.4   Added Constraints in R22-11

Number	Heading
[constr_5340]	Range of `DltMessage.privacyLevel.privacyLevel`
[constr_5341]	Range of `PrivacyLevel.compuMethod`

Table A.3: Added Constraints in R22-11

A.2.5   Changed Constraints in R22-11

Number	Heading
[constr_5098]	Allowed `SwDataDefProps` attributes for `DltArgument.networkRepresentation`
[constr_5302]	Restriction in usage of `DltArgument.optional` attribute

Table A.4: Changed Constraints in R22-11

A.2.6   Deleted Constraints in R22-11

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
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
fileInfo Comment	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	BaseTypeDirectDefinition			
Package	M2::MSR::AsamHdo::BaseTypes			
Note	This BaseType is defined directly (as opposite to a derived BaseType)			
Base	ARObject, BaseTypeDefinition			
Aggregated by	BaseType.baseTypeDefinition			
Attribute	Type	Mult.	Kind	Note
baseType Encoding	BaseTypeEncoding String	0..1	attr	This specifies, how an object of the current BaseType is encoded, e.g. in an ECU within a message sequence. **Tags:**xml.sequenceOffset=90
baseTypeSize	PositiveInteger	0..1	attr	Describes the length of the data type specified in the container in bits. **Tags:**xml.sequenceOffset=70
byteOrder	ByteOrderEnum	0..1	attr	This attribute specifies the byte order of the base type. **Tags:**xml.sequenceOffset=110

$\triangledown$

△

Class	BaseTypeDirectDefinition			
memAlignment	PositiveInteger	0..1	attr	This attribute describes the alignment of the memory object in bits. E.g. "8" specifies, that the object in question is aligned to a byte while "32" specifies that it is aligned four byte. If the value is set to "0" the meaning shall be interpreted as "unspecified". **Tags:**xml.sequenceOffset=100
native Declaration	NativeDeclarationString	0..1	attr	This attribute describes the declaration of such a base type in the native programming language, primarily in the Programming language C. This can then be used by a code generator to include the necessary declarations into a header file. For example BaseType with shortName: "MyUnsignedInt" native Declaration: "unsigned short" Results in typedef unsigned short MyUnsignedInt; If the attribute is not defined the referring Implementation DataTypes will not be generated as a typedef by RTE. If a nativeDeclaration type is given it shall fulfill the characteristic given by basetypeEncoding and baseType Size. This is required to ensure the consistent handling and interpretation by software components, RTE, COM and MCM systems. **Tags:**xml.sequenceOffset=120

Table B.2: BaseTypeDirectDefinition

Class	CompuMethod			
Package	M2::MSR::AsamHdo::ComputationMethod			
Note	This meta-class represents the ability to express the relationship between a physical value and the mathematical representation. Note that this is still independent of the technical implementation in data types. It only specifies the formula how the internal value corresponds to its physical pendant. **Tags:**atp.recommendedPackage=CompuMethods			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *CollectableElement*, *Identifiable*, *Multilanguage Referrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
compuInternal ToPhys	Compu	0..1	aggr	This specifies the computation from internal values to physical values. **Tags:**xml.sequenceOffset=80
compuPhysTo Internal	Compu	0..1	aggr	This represents the computation from physical values to the internal values. **Tags:**xml.sequenceOffset=90
displayFormat	DisplayFormatString	0..1	attr	This property specifies, how the physical value shall be displayed e.g. in documents or measurement and calibration tools. **Tags:**xml.sequenceOffset=20

▽

△

Class	CompuMethod			
unit	Unit	0..1	ref	This is the physical unit of the Physical values for which the CompuMethod applies. **Tags:**xml.sequenceOffset=30

Table B.3: CompuMethod

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*			
Subclasses	ARPackage, *AbstractDoIpLogicAddressProps*, *AbstractEvent*, *AbstractImplementationDataTypeElement*, *AbstractSecurityEventFilter*, *AbstractSecurityIdsmInstanceFilter*, *AbstractServiceInstance*, Application Endpoint, ApplicationError, ArtifactChecksum, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpFeature*, AutosarOperationArgumentInstance, AutosarVariableInstance, BlockState, *BuildActionEntity*, BuildAction Environment, Chapter, ClassContentConditional, ClientIdDefinition, ClientServerOperation, Code, *CollectableElement*, ComManagementMapping, *CommConnectorPort*, *CommunicationConnector*, *CommunicationController*, Compiler, ConsistencyNeeds, ConsumedEventGroup, CouplingPort, *Coupling PortStructuralElement*, CryptoKeySlot, *CryptoServiceMapping*, DataPrototypeGroup, Data Transformation, DependencyOnArtifact, *DiagEventDebounceAlgorithm*, DiagnosticConnectedIndicator, DiagnosticDataElement, DiagnosticDebounceAlgorithmProps, DiagnosticFunctionInhibitSource, DiagnosticParameterElement, *DiagnosticRoutineSubfunction*, DltApplication, DltArgument, DltMessage, DoIpInterface, DoIpLogicAddress, DoIpRoutingActivation, EndToEndProtection, EthernetWakeupSleep OnDatalineConfig, EventHandler, ExclusiveArea, *ExecutableEntity*, *ExecutionTime*, FMAttributeDef, FM FeatureMapAssertion, FMFeatureMapCondition, FMFeatureMapElement, FMFeatureRelation, FM FeatureRestriction, FMFeatureSelection, FlexrayArTpNode, FlexrayTpPduPool, *FrameTriggering*, GeneralParameter, GlobalTimeGateway, *GlobalTimeMaster*, *GlobalTimeSlave*, *HeapUsage*, HwAttribute Def, HwAttributeLiteralDef, HwPin, HwPinGroup, IPSecRule, IPv6ExtHeaderFilterList, ISignalToIPdu Mapping, ISignalTriggering, *IdentCaption*, InternalTriggeringPoint, Keyword, LifeCycleState, Linker, Mac MulticastGroup, MacSecKayParticipant, McDataInstance, MemorySection, ModeDeclaration, Mode DeclarationMapping, ModeSwitchPoint, NetworkEndpoint, *NmCluster*, *NmNode*, *PackageableElement*, ParameterAccess, PduActivationRoutingGroup, PduToFrameMapping, PduTriggering, PerInstance Memory, *PhysicalChannel*, PortGroup, *PortInterfaceMapping*, PossibleErrorReaction, Resource Consumption, RootSwCompositionPrototype, RptComponent, RptContainer, RptExecutableEntity, Rpt ExecutableEntityEvent, RptExecutionContext, RptProfile, RptServicePoint, RunnableEntityGroup, *Sdg Attribute*, SdgClass, SecureCommunicationAuthenticationProps, SecureCommunicationFreshnessProps, SecurityEventContextProps, *ServiceNeeds*, SignalServiceTranslationEventProps, SignalService TranslationProps, SocketAddress, SomeipTpChannel, *SpecElementReference*, *StackUsage*, Static SocketConnection, StructuredReq, SwGenericAxisParamType, SwServiceArg, SwcServiceDependency, SystemMapping, *TimeBaseResource*, *TimingClock*, TimingClockSyncAccuracy, TimingCondition, *Timing Constraint*, *TimingDescription*, TimingExtensionResource, TimingModeInstance, Topic1, TpAddress, TraceableTable, TraceableText, *TracedFailure*, *TransformationProps*, TransformationTechnology, Trigger, VariableAccess, VariationPointProxy, ViewMap, VlanConfig, WaitPoint			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
adminData	AdminData	0..1	aggr	This represents the administrative data for the identifiable object. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=adminData 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

▽

△

Class	Identifiable (abstract)			
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
desc	MultiLanguageOverview Paragraph	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
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 B.4: Identifiable

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			
Aggregated by	AtpClassifier.atpFeature, SwComponentType.port			
Attribute	Type	Mult.	Kind	Note
–	–	–	–	–

Table B.5: PortPrototype

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			
Subclasses	AtpDefinition, BswDistinguishedPartition, BswModuleCallPoint, BswModuleClientServerEntry, Bsw VariableAccess, CouplingPortTrafficClassAssignment, DiagnosticEnvModeElement, EthernetPriority Regeneration, ExclusiveAreaNestingOrder, HwDescriptionEntity, ImplementationProps, ModeTransition, MultilanguageReferrable, PncMappingIdent, SingleLanguageReferrable, SoConIPduIdentifier, Socket ConnectionBundle, TimeSyncServerConfiguration, TpConnectionIdent			
Attribute	Type	Mult.	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. **Stereotypes:** atpIdentityContributor **Tags:** xml.enforceMinMultiplicity=true xml.sequenceOffset=-100
shortName Fragment	ShortNameFragment	*	aggr	This specifies how the Referrable.shortName is composed of several shortNameFragments. **Tags:**xml.sequenceOffset=-90

Table B.6: Referrable

Class	SwBaseType			
Package	M2::MSR::AsamHdo::BaseTypes			
Note	This meta-class represents a base type used within ECU software. **Tags:**atp.recommendedPackage=BaseTypes			
Base	ARElement, ARObject, AtpBlueprint, AtpBlueprintable, BaseType, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note
–	–	–	–	–

Table B.7: SwBaseType

Class	<<atpVariation>> SwDataDefProps			
Package	M2::MSR::DataDictionary::DataDefProperties			
Note	This class is a collection of properties relevant for data objects under various aspects. One could consider this class as a "pattern of inheritance by aggregation". The properties can be applied to all objects of all classes in which SwDataDefProps is aggregated. **Tags:**vh.latestBindingTime=codeGenerationTime			
Base	ARObject			
Aggregated by	AutosarDataType.swDataDefProps, CompositeNetworkRepresentation.networkRepresentation, Data Prototype.swDataDefProps, DataPrototypeTransformationProps.networkRepresentationProps, DiagnosticDataElement.swDataDefProps, DiagnosticEnvDataElementCondition.swDataDefProps, Dlt Argument.networkRepresentation, FlatInstanceDescriptor.swDataDefProps, ImplementationDataType Element.swDataDefProps, InstantiationDataDefProps.swDataDefProps, ISignal.networkRepresentation Props, McDataInstance.resultingProperties, ParameterAccess.swDataDefProps, PerInstanceMemory.sw DataDefProps, ReceiverComSpec.networkRepresentation, SenderComSpec.networkRepresentation, SomeipDataPrototypeTransformationProps.networkRepresentation, SwPointerTargetProps.swDataDef Props, SwServiceArg.swDataDefProps, SwSystemconst.swDataDefProps, SystemSignal.physicalProps			
Attribute	Type	Mult.	Kind	Note

▽

△

Class	<<atpVariation>> **SwDataDefProps**			
annotation	Annotation	*	aggr	This aggregation allows to add annotations (yellow pads ...) related to the current data object. **Tags:** xml.roleElement=true xml.roleWrapperElement=true xml.sequenceOffset=20 xml.typeElement=false xml.typeWrapperElement=false
baseType	SwBaseType	0..1	ref	Base type associated with the containing data object. **Tags:**xml.sequenceOffset=50
compuMethod	CompuMethod	0..1	ref	Computation method associated with the semantics of this data object. **Tags:**xml.sequenceOffset=180
dataConstr	DataConstr	0..1	ref	Data constraint for this data object. **Tags:**xml.sequenceOffset=190
displayFormat	DisplayFormatString	0..1	attr	This property describes how a number is to be rendered e.g. in documents or in a measurement and calibration system. **Tags:**xml.sequenceOffset=210
display Presentation	DisplayPresentation Enum	0..1	attr	This attribute controls the presentation of the related data for measurement and calibration tools.
invalidValue	ValueSpecification	0..1	aggr	Optional value to express invalidity of the actual data element. **Tags:**xml.sequenceOffset=255
swComparison Variable	SwVariableRefProxy	*	aggr	Variables used for comparison in an MCD process. **Tags:** xml.sequenceOffset=170 xml.typeElement=false
swHostVariable	SwVariableRefProxy	0..1	aggr	Contains a reference to a variable which serves as a host-variable for a bit variable. Only applicable to bit objects. **Tags:** xml.sequenceOffset=220 xml.typeElement=false
swTextProps	SwTextProps	0..1	aggr	the specific properties if the data object is a text object. **Tags:**xml.sequenceOffset=120
unit	Unit	0..1	ref	Physical unit associated with the semantics of this data object. This attribute applies if no compuMethod is specified. If both units (this as well as via compuMethod) are specified the units shall be compatible. **Tags:**xml.sequenceOffset=350

Table B.8: SwDataDefProps