Document Title	Specification of ECU Configuration
Document Category	Template specification (TPS)
Document UID	87
Document Status	published
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	R26-11

Document Change History

Date	Release	Changed by	Description
2026-11-26	R26-11	AUTOSAR Release Management	• PostBuild ConfigClass simplification [CONC 739] • Simplify Uri Reference (remove destinationUriNestingContract) • Editorial: Move chapter on service needs from appendix to main text • Editorial: Introduce imposition times for constraints • Minor changes
2025-11-27	R25-11	AUTOSAR Release Management	• Minor changes • Editorial: Replace `ECU Configuration Value Description` by `ECU Configuration Values` or `ECU Configuration Values ARXML` • Editorial: Improved descriptions of URI References Chapter 2.3.6.6 • Added BusToPartitionMapping Chapter 3.3.6
2024-11-27	R24-11	AUTOSAR Release Management	• Added EcucPartitionId and EcucPartitionCoreRef to Ecuc module • Added configuration for Complex Drivers, which interact with the L-Sdu Router module • Added Pdu Meta-Data used for IEEE1722Tp • Changes in specification items and constraints: for details please see the change history

△

2023-11-23	R23-11	AUTOSAR Release Management	• Minor changes
2022-11-24	R22-11	AUTOSAR Release Management	• Clarified usage of apiServicePrefix for Xfrm module defintions • Added origin flag to EcucContainerDef • Improved description how optionality of containers, parameters and references in the EcuC Parameter Definition UML model is expressed • Clarified semantic of multiple aggregated container trees that include references
2021-11-25	R21-11	AUTOSAR Release Management	• Extend `EcucParameterDef`s with `symbolicNameValue` to support `PublishedInformation`. • Added `withAuto` support to `EcucAbstractReferenceDef`.
2020-11-30	R20-11	AUTOSAR Release Management	• Introduced CddModuleId parameter • Removed “Rules for Configuration Editors” chapter • Changed all lower multiplicities in the Ecuc meta-model to 0 and introduced constraints that define at which time which model elements need to be available. For details please refer to the ChangeDocumentation.
2019-11-28	R19-11	AUTOSAR Release Management	• Updated specification to avoid usage of term MUST. • Specification of the format of the destinationType of an EcucForeignReferenceDef • Added support for Bsw Multicore Distribution in Ecuc module • Changed Document Status from Final to published

▽

△

2018-10-31	4.4.0	AUTOSAR Release Management	• Removed `EcucSymbolicNameReferenceDef` • Introduced postBuildVariantsUsed flag to improve the configuration of postBuild variants • Minor corrections / clarifications / editorial changes; For details please refer to the ChangeDocumentation
2017-12-08	4.3.1	AUTOSAR Release Management	• Minor corrections / clarifications / editorial changes; For details please refer to the ChangeDocumentation
2016-11-30	4.3.0	AUTOSAR Release Management	• Minor corrections / clarifications / editorial changes; For details please refer to the ChangeDocumentation
2015-07-31	4.2.2	AUTOSAR Release Management	• Minor corrections / clarifications / editorial changes; For details please refer to the ChangeDocumentation
2014-10-31	4.2.1	AUTOSAR Release Management	• Improved description of Post-build variants • Improved Post-build loadable approach • Introduction of Uri References • Minor corrections / clarifications / editorial changes; For details please refer to the BWCStatement
2014-03-31	4.1.3	AUTOSAR Release Management	• Various fixes and clarifications
2013-10-31	4.1.2	AUTOSAR Release Management	• Support unidirectional CDD communication • Adapted range of parameter MetaDataLength • Harmonization with TR_Methodology • Added "origin" attribute to the EcucContainerDef • Adapted CDD configuration to allow the configuration of the CDD interface type (IF/TP) • Adapted the upper limit of PduLength parameter

▽

△

			△ • Stereotyped EcucChoiceReferenceDef.destination and EcucSymbolicNameReference-Def.destination with atpUriDef
2013-03-15	4.1.1	AUTOSAR Administration	• Description of the variant handling approach to cope with PreCompile, Link and Post-Build Configuration parameters as alternative to the usage of multiple configuration containers • Made the CDD configuration postBuildConfigurable • Updated sorting criteria for EcucContainerValues • Extended CDD configuration with SoAd interaction • Clarified the production error configuration • The destination of EcucReferenceDef and EcucChoiceReferenceDef is changed to EcucContainerDef • Extended the Ecu Query Language to describe configuration validity rules • Added apiServicePrefix attribute to EcucModuleDef • Added EcucPartitionBswModuleExecution and EcucPartitionBswModuleDistinguished-Partition • Updated section about the conversion of time parameters of main functions to ticks • Added EcucCoreDefinition to Ecuc module

▽

△

2011-12-22	4.0.3	AUTOSAR Administration	• ecuc_sws_5001 removed. • Clarified modeling of destinationType and destinationContext. • Clarified scope of parameters. • Clarified postBuildChangeable and multipleConfigurationContainer. • Added annotation to EcucAbstractReferenceValue. • Updated semantics of definitionRef and introduced the term "pure VSMD" • Clarification of PostBuildSelectable, PostBuildLoadable in VSMD • Set configuration class affection support to deprecated • Support for ordering of EcucParameters and EcucReferences • Reworked CDD configuration to reflect the direction of the communication • Clarified usage of symbolic name references
2011-04-15	4.0.2	AUTOSAR Administration	• Updated "refvalue" function requirements • Added requirement sws6045 • Changed specification of PduLength parameter from bits to bytes • Added attribute "origin" to EcucEnumerationParamDef • Added "Template Glossary" to Appendix • Added "Rules for navigating in Ecu Configuration Artifacts" chapter • Removed restriction on hex-representation of integers • Updated description of refinedModuleDef within class ModuleDef • Changed calculation language key words to lower case • Changed structure of EcucQuery and EcucQueryExpression

▽

			• Added section on Communication Channel ID • Removed section on EcucMemoryMappingCollection • Removed "annotation" from "EcucContainerValue"
2009-12-18	4.0.1	AUTOSAR Administration	• Implemented Variant Handling concept • Implemented Calculation Formula concept • Reworked Parameter Value representation • Reworked Service Component Methodology chapter • Updated rules for deriving VSMD from StMD • Implemented Documentation support concept • Implemented support for existence dependence of EcuC Parameter Definition elements • Added "Clock Tree Configuration" chapter • Added "CDD module" chapter
2009-02-04	3.1.2	AUTOSAR Administration	• Fixed foreign reference to PduToFrameMapping
2008-08-13	3.1.1	AUTOSAR Administration	• Added reference from Container to ContainerDef. • Removed reference from Container to ParamConfContainerDef.
2007-12-21	3.0.1	AUTOSAR Administration	• Changed representation of a ChoiceContainerDef in an ECU Configuration Description • Moved sections from "ECU Configuration Parameter Definition" into the "Specification of ECU Configuration" (COM-Stack Configuration Patterns) • Updated interaction of ECU Configuration with BSW Module Description

△

			△ • Added specification items which define what is allowed when creating a Vendor Specific Module Definition (VSMD) • Correction of "InstanceParamRef" definition in ECU Configuration Specification • Refined the available character set of calculationFormula • Added clarification about the usage of ADMIN-DATA to track version information • Document meta information extended • Small layout adaptations made
2007-01-24	2.1.15	AUTOSAR Administration	• “Advice for users” revised • Legal disclaimer revised
2006-11-28	2.1	AUTOSAR Administration	• Methodology chapter revised (incl. introduction of support for AUTOSAR Services) • Added EcucElement, EcuSwComposition, configuration class affection, LinkerSymbolDef and LinkerSymbolValue to the metamodel • Support for multiple configuration sets added • Legal disclaimer revised
2006-05-16	2.0	AUTOSAR Administration	• 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.

Table of Contents

1 Introduction 16
1.1 Document Conventions . . . 18
2 Configuration Meta-model 21
2.1 Introduction . . . 21
2.2 ECU Configuration Template Structure . . . 21
2.3 ECU Configuration Parameter Definition Meta-model . . . 26
2.3.1 ECU Configuration Parameter Definition top-level structure . . . 26
2.3.1.1 Usage of the Admin Data . . . 28
2.3.1.2 Life Cycle definition . . . 29
2.3.1.3 Documentation Support . . . 31
2.3.2 ECU Configuration Module Definition . . . 33
2.3.3 Container Definition . . . 38
2.3.3.1 Choice Container Definition . . . 43
2.3.4 Common Configuration Elements . . . 46
2.3.4.1 Variant Handling . . . 46
2.3.4.2 Configuration Multiplicity . . . 46
2.3.4.3 Common Configuration Attributes . . . 49
2.3.5 Parameter Definition . . . 57
2.3.5.1 Boolean Type . . . 59
2.3.5.2 Integer Type . . . 61
2.3.5.3 Float Type . . . 62
2.3.5.4 String Parameter . . . 64
2.3.5.5 Linker Symbol Parameter . . . 66
2.3.5.6 Function Name Parameter . . . 67
2.3.5.7 Enumeration Parameter . . . 67
2.3.5.8 Enumeration Literal Definition . . . 68
2.3.5.9 AddInfo . . . 69
2.3.6 References in Parameter Definition . . . 70
2.3.6.1 Reference . . . 74
2.3.6.2 Choice Reference . . . 75
2.3.6.3 Foreign Reference . . . 77
2.3.6.4 Instance Reference . . . 78
2.3.6.5 Symbolic Name Reference . . . 81
2.3.6.6 Uri Reference . . . 83
2.3.7 Derived Parameter Specification . . . 92
2.3.7.1 Derived Parameter Calculation Formula . . . 94
2.3.7.2 Restrictions on Configuration Class of Derived Parameters . . . 105
2.3.8 Existence dependence of ECUC Parameter Definition elements . . . 105
2.3.9 Validation conditions . . . 109
2.3.10 Multiple aggregation of full container trees that include references . . . 110
2.4 ECU Configuration Value Meta-model . . . 114

2.4.1 ECU Configuration Value Top-Level Structure . . . 114
2.4.2 Module Configurations . . . 116
2.4.2.1 Splitable ModuleConfiguration . . . 121
2.4.3 Parameter Container Description . . . 123
2.4.3.1 Choice Containers . . . 128
2.4.4 Parameter Values . . . 130
2.4.4.1 Textual Parameter Value . . . 133
2.4.4.2 Numerical Parameter Value . . . 135
2.4.4.3 AddInfo Parameter Value . . . 136
2.4.5 References in the ECU Configuration Meta-model . . . 136
2.4.5.1 Instance Reference Values . . . 141
2.4.5.2 Representation of Symbolic Names . . . 142
2.4.5.3 Rules for references in the ECUC Configuration Values . . . 149
2.4.6 Derived Parameters in ECU Configuration Values . . . 150
2.4.7 Using Variant Handling to Cope with Several Binding Times in the ECU Configuration Values . . . 152
2.4.7.1 Example of ECU configuration using Variant Handling . . . 153
3 ECU Configuration Parameter Definition SWS implications 160
3.1 Formalization aspects . . . 160
3.1.1 ECU Configuration Parameter Definition table . . . 161
3.2 AUTOSAR Stack Overview . . . 163
3.3 Virtual Module EcuC . . . 169
3.3.1 Hardware description . . . 170
3.3.2 Definition of Partitions . . . 172
3.3.3 PostBuild Variants . . . 177
3.3.4 Variation Resolver Description . . . 178
3.3.5 UnitGroup Assignment . . . 180
3.3.6 Bus to Partition Mapping . . . 181
3.3.7 Definition of Pdus . . . 184
3.3.8 Pdu Meta-Data . . . 191
3.4 COM-Stack configuration . . . 198
3.4.1 Handle IDs . . . 198
3.4.1.1 Handle ID concept . . . 199
3.4.1.2 Definition of Handle IDs . . . 200
3.4.1.3 Agreement on Handle IDs . . . 201
3.4.1.4 Handle IDs with symbolic names . . . 202
3.4.2 Configuration examples for the Pdu Router . . . 204
3.4.2.1 Tx from Com to CanIf . . . 204
3.4.2.2 Rx from CanIf to Com . . . 205
3.4.2.3 Gateway from CanIf to FrIf . . . 206
3.4.3 Communication Channel IDs . . . 206
3.4.4 Copy Configuration for PDUs . . . 207
3.5 CDD module . . . 208

3.5.1 Pdu Router . . . 213
3.5.2 L-Sdu Router . . . 222
3.5.3 COM Interface modules . . . 229
3.5.4 Communication Manager . . . 232
3.5.5 Generic Network Management . . . 233
3.5.6 Socket Adaptor . . . 235
3.5.7 J1939Rm . . . 240
3.5.8 Global Time Synchronization . . . 242
3.6 EcuM configuration to initialize post-build capable BSW Modules . . . 247
3.7 Optional reporting of Production Errors and Extended Production Errors 247
3.8 Converting time parameters of main functions to ticks . . . 248
3.9 Clock Tree Configuration . . . 249
4 Rules to follow in different configuration activities 252
4.1 Deriving vendor specific module definitions from standardized module definitions . . . 252
4.2 Rules for building the Base ECU configuration . . . 265
4.3 Rules for navigating in ECU Configuration Artifacts . . . 267
4.4 Post-build Time Consistency . . . 268
4.5 Impact of ServiceNeeds on ECU Configuration . . . 268
A Reference Material 271
A.1 Abbreviations . . . 271
A.2 Imposition Times of Constraints . . . 271
B Possible Implementations for the Configuration Steps 273
B.1 Alternative Approaches (Informative) . . . 273
B.1.1 Alternative Configuration Editor Approaches . . . 273
B.1.1.1 Custom Editors . . . 274
B.1.1.2 Generic Tools . . . 275
B.1.1.3 Tools Framework . . . 276
B.1.2 Alternative Generation Approaches . . . 276
C Glossary 278
D Change History 281
D.1 Change History for release R4.0.1 . . . 281
D.1.1 Renamed Meta-Model Elements for AUTOSAR Release 4.0 . . . 281
D.1.2 Deleted SWS Items . . . 281
D.1.3 Changed SWS Items . . . 282
D.1.4 Added SWS Items . . . 283
D.2 Change History for release R4.0.2 . . . 285
D.2.1 Changed SWS Items . . . 285
D.2.2 Added SWS Items . . . 285
D.3 Change History for release R4.0.2 . . . 286
D.3.1 Deleted SWS Items . . . 286

D.3.2 Changed SWS Items . . . 286
D.3.3 Added SWS Items . . . 287
D.3.4 Added Constraints . . . 288
D.4 Change History for release R4.1.1 . . . 288
D.4.1 Deleted SWS Items . . . 288
D.4.2 Changed SWS Items . . . 288
D.4.3 Added SWS Items . . . 289
D.4.4 Added Constraints . . . 290
D.5 Change History for release R4.1.2 . . . 290
D.5.1 Deleted SWS Items . . . 290
D.5.2 Changed SWS Items . . . 290
D.5.3 Added SWS Items . . . 291
D.6 Change History for release R4.1.3 . . . 291
D.6.1 Deleted SWS Items . . . 291
D.6.2 Changed SWS Items . . . 291
D.6.3 Added SWS Items . . . 291
D.6.4 Added Constraints . . . 292
D.7 Change History for release R4.2.1 . . . 292
D.7.1 Added Specification Items in 4.2.1 . . . 292
D.7.2 Changed Specification Items in 4.2.1 . . . 294
D.7.3 Deleted Specification Items in 4.2.1 . . . 294
D.7.4 Added Constraints in 4.2.1 . . . 295
D.7.5 Changed Constraints in 4.2.1 . . . 296
D.7.6 Deleted Constraints in 4.2.1 . . . 296
D.8 Change History for release R4.2.2 . . . 296
D.8.1 Added Specification Items in 4.2.2 . . . 296
D.8.2 Changed Specification Items in 4.2.2 . . . 297
D.8.3 Deleted Specification Items in 4.2.2 . . . 297
D.8.4 Added Constraints in 4.2.2 . . . 297
D.8.5 Changed Constraints in 4.2.2 . . . 297
D.8.6 Deleted Constraints in 4.2.2 . . . 298
D.9 Change History for release R4.3.0 . . . 298
D.9.1 Added Specification Items in 4.3.0 . . . 298
D.9.2 Changed Specification Items in 4.3.0 . . . 298
D.9.3 Deleted Specification Items in 4.3.0 . . . 298
D.9.4 Added Constraints in 4.3.0 . . . 298
D.9.5 Changed Constraints in 4.3.0 . . . 299
D.9.6 Deleted Constraints in 4.3.0 . . . 299
D.10 Change History for release R4.3.1 . . . 299
D.10.1 Added Specification Items in 4.3.1 . . . 299
D.10.2 Changed Specification Items in 4.3.1 . . . 299
D.10.3 Deleted Specification Items in 4.3.1 . . . 299
D.10.4 Added Constraints in 4.3.1 . . . 300

D.10.5 Changed Constraints in 4.3.1 . . . 300
D.10.6 Deleted Constraints in 4.3.1 . . . 300
D.11 Change History for release R4.4.0 . . . 300
D.11.1 Added Specification Items in 4.4.0 . . . 300
D.11.2 Changed Specification Items in 4.4.0 . . . 300
D.11.3 Deleted Specification Items in 4.4.0 . . . 301
D.11.4 Added Constraints in 4.4.0 . . . 301
D.11.5 Changed Constraints in 4.4.0 . . . 301
D.11.6 Deleted Constraints in 4.4.0 . . . 301
D.12 Change History for release R19-11 . . . 302
D.12.1 Added Specification Items in 19-11 . . . 302
D.12.2 Changed Specification Items in 19-11 . . . 302
D.12.3 Deleted Specification Items in 19-11 . . . 302
D.12.4 Added Constraints in 19-11 . . . 302
D.12.5 Changed Constraints in 19-11 . . . 303
D.12.6 Deleted Constraints in 19-11 . . . 303
D.13 Change History for release R20-11 . . . 303
D.13.1 Added Specification Items in R20-11 . . . 303
D.13.2 Changed Specification Items in R20-11 . . . 303
D.13.3 Deleted Specification Items in R20-11 . . . 303
D.13.4 Added Constraints in R20-11 . . . 304
D.13.5 Changed Constraints in R20-11 . . . 304
D.13.6 Deleted Constraints in R20-11 . . . 305
D.14 Change History for release R21-11 . . . 305
D.14.1 Added Specification Items in R21-11 . . . 305
D.14.2 Changed Specification Items in R21-11 . . . 305
D.14.3 Deleted Specification Items in R21-11 . . . 305
D.14.4 Added Constraints in R21-11 . . . 305
D.14.5 Changed Constraints in R21-11 . . . 305
D.14.6 Deleted Constraints in R21-11 . . . 305
D.15 Change History for release R22-11 . . . 306
D.15.1 Added Specification Items in R22-11 . . . 306
D.15.2 Changed Specification Items in R22-11 . . . 306
D.15.3 Deleted Specification Items in R22-11 . . . 306
D.15.4 Added Constraints in R22-11 . . . 306
D.15.5 Changed Constraints in R22-11 . . . 307
D.15.6 Deleted Constraints in R22-11 . . . 307
D.16 Change History for release R23-11 . . . 307
D.16.1 Added Specification Items in R23-11 . . . 307
D.16.2 Changed Specification Items in R23-11 . . . 307
D.16.3 Deleted Specification Items in R23-11 . . . 308
D.16.4 Added Constraints in R23-11 . . . 308
D.16.5 Changed Constraints in R23-11 . . . 308

D.16.6 Deleted Constraints in R23-11 . . . 308
D.17 Change History for release R24-11 . . . 308
D.17.1 Added Specification Items in R24-11 . . . 308
D.17.2 Changed Specification Items in R24-11 . . . 308
D.17.3 Deleted Specification Items in R24-11 . . . 309
D.17.4 Added Constraints in R24-11 . . . 309
D.17.5 Changed Constraints in R24-11 . . . 309
D.17.6 Deleted Constraints in R24-11 . . . 309
D.18 Change History for release R25-11 . . . 310
D.18.1 Added Specification Items in R25-11 . . . 310
D.18.2 Changed Specification Items in R25-11 . . . 310
D.18.3 Deleted Specification Items in R25-11 . . . 310
D.18.4 Added Constraints in R25-11 . . . 311
D.18.5 Changed Constraints in R25-11 . . . 311
D.18.6 Deleted Constraints in R25-11 . . . 311
D.19 Change History for release R26-11 . . . 312
D.19.1 Added Specification Items in R26-11 . . . 312
D.19.2 Changed Specification Items in R26-11 . . . 314
D.19.3 Deleted Specification Items in R26-11 . . . 314
D.19.4 Added Constraints in R26-11 . . . 316
D.19.5 Changed Constraints in R26-11 . . . 317
D.19.6 Deleted Constraints in R26-11 . . . 317
E Mentioned Class Tables 319
F Non-Splitable Elements in the Scope of this Document 350
G Variation Points in the Scope of this Document 351

References

[1] System Template
AUTOSAR_CP_TPS_SystemTemplate

[2] Methodology for Classic Platform
AUTOSAR_CP_TR_Methodology

[3] Standardization Template
AUTOSAR_FO_TPS_StandardizationTemplate

[4] Generic Structure Template
AUTOSAR_FO_TPS_GenericStructureTemplate

[5] AUTOSAR XML Schema Production Rules
AUTOSAR_FO_TPS_XMLSchemaProductionRules

[6] Glossary
AUTOSAR_FO_TR_Glossary

[7] Specification of ECU Configuration Parameters (XML)
AUTOSAR_CP_MOD_ECUConfigurationParameters

[8] Meta Model
AUTOSAR_FO_MMOD_MetaModel

[9] IEEE standard for radix-independent floating-point arithmetic
(ANSI/IEEE Std 854-1987)

[10] Meta Model-generated XML Schema
AUTOSAR_FO_MMOD_XMLSchema

[11] Software Component Template
AUTOSAR_CP_TPS_SoftwareComponentTemplate

[12] Layered Software Architecture
AUTOSAR_CP_EXP_LayeredSoftwareArchitecture

[13] Basic Software Module Description Template
AUTOSAR_CP_TPS_BSWModuleDescriptionTemplate

[14] Specification of ECU Configuration
AUTOSAR_CP_TPS_ECUConfiguration

[15] Guide to BSW Distribution
AUTOSAR_CP_EXP_BSWDistributionGuide

[16] General Specification of Basic Software Modules
AUTOSAR_CP_SWS_BSWGeneral

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

1 Introduction

According to the AUTOSAR Methodology (see figure 1.1) the configuration process is a major part of the ECU software integration that is represented by the activity `Integrate Software for ECU`.

The configuration process of an ECU starts with the splitting of the System Description into several descriptions, whereas each contains all information about one single ECU. In figure 1.1 the artifact `System Description` is hidden in the activity `Develop System`. The creation of an `Ecu Extract` is described in detail in the System Template specification [1].

The `Ecu Extract` and the `BSW Module Delivered Bundle` are the inputs for the ECU configuration step. This is also visible in figure 1.2 where the ECU configuration is described by the activities `Prepare ECU Configuration` and `Configure BSW and RTE`.

A detailed description about these activities is given in the AUTOSAR Methodology [2] Chapter 2.7 *"Integrate Software for ECU"*.

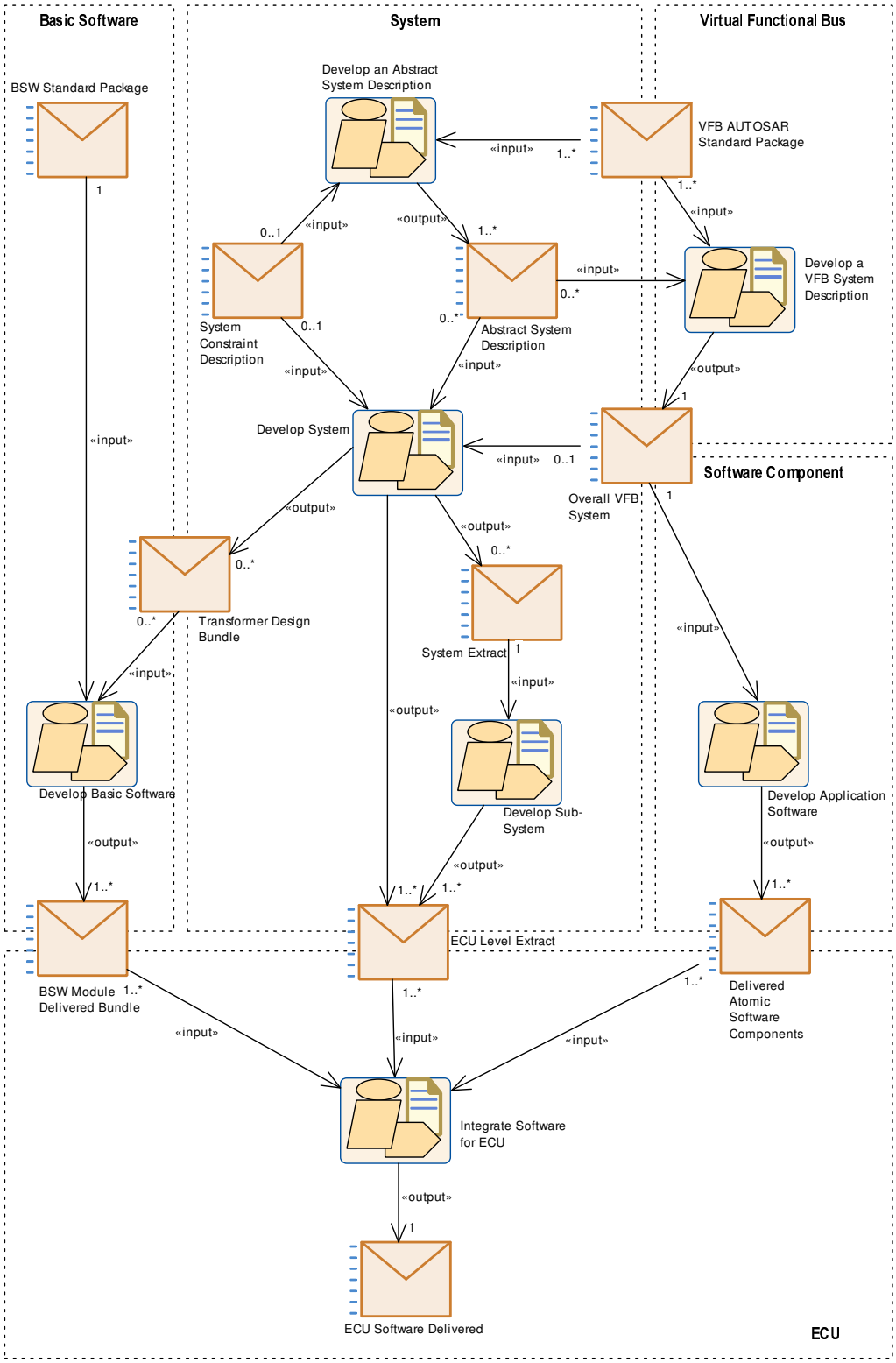

Figure 1.1: AUTOSAR Methodology Overview (from [2])

Within the ECU Configuration process each single module of the AUTOSAR Architecture can be configured for the special needs of this ECU. Because of a quite complex AUTOSAR Architecture, modules and interdependencies between the modules, tool-support is required: AUTOSAR ECU Configuration Editor(s).

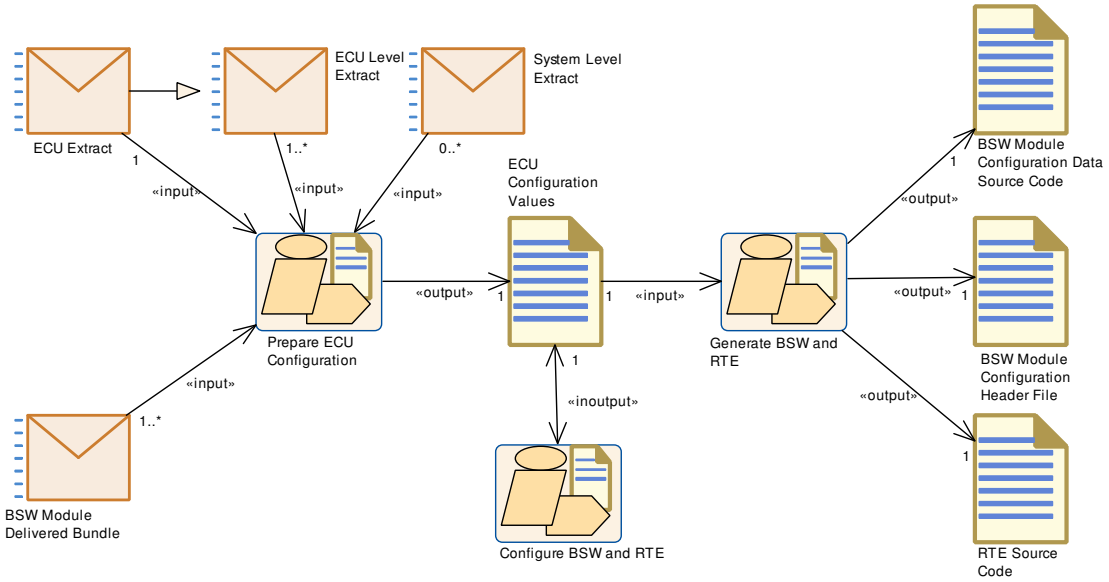

Figure 1.2: ECU Configuration Overview (from [2])

The tool strategy and tooling details for the ECU Configuration are out of scope of this specification. Nevertheless tools need the knowledge about ECU Configuration Parameters and their constraints such as configuration class, value range, multiplicities etc. This description is the input for the tools. The description of configuration parameters is called `ECU Configuration Parameter Definition` and described in detail in this specification (chapter 2.3).

To make sure that all tools are using the same output-format within the configured values of the parameters, the `ECU Configuration Values` is also part of this specification and described in detail later (chapter 2.4). The `ECU Configuration Values` may be on one hand the input format for other configuration tools (within a tool-chain of several configuration editors) and on the other hand it is the basis of generators. The configured parameters are generated into ECU executables. This is the last step of the configuration process and again out of scope of this specification.

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 “imposition 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 imposition times of constraints are defined and the imposition 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 [3, Standardization Template].

Please note: By intent, TPS documents (and their traceable items) do not trace up to an RS (requirement item).

2 Configuration Meta-model

2.1 Introduction

AUTOSAR exchange formats are specified using a meta-model based approach. The meta-model for the configuration of ECU artifacts uses an universal description language so that it is possible to specify different kinds of configuration aspects. This is important as it is possible to describe AUTOSAR-standardized and vendor-specific ECU Configuration Parameters with the same set of language elements. This eases the development of tools and introduces the possibility to standardize vendor-specific ECU Configuration Parameters at a later point in time.

In general the configuration language uses containers and actual parameters. Containers are used to group corresponding parameters. Parameters hold the relevant values that configure the specific parts of an ECU. Due to the flexibility that has to be achieved by the configuration language the configuration description is divided into two parts:

- `ECU Configuration Parameter Definition`
- `ECU Configuration Values`

A detailed description of these two parts and their relationships is presented in the following sections.

2.2 ECU Configuration Template Structure

In this section the relationships between the different AUTOSAR templates involved in the ECU Configuration are introduced. A template is defining the structure and possible content of an actual description. The concept is open to be implemented in several possible ways, in AUTOSAR XML files have been chosen to be used for the exchange formats. If XML files are used there is no conceptual limit in the number of files making up the description. All the contributing files are virtually merged to build the actual description.

The goal of the `ECU Configuration Value Template` is to specify an exchange format for the `ECU Configuration Values` of one ECU. The actual output of ECU Configuration editors is stored in the `ECU Configuration Values`, which might be one or several XML files. But the ECU Configuration editors need to know how the content of an `ECU Configuration Values` should be structured (which parameters are available in which container) and what kind of restrictions are to be respected (e.g. the ECU Configuration Parameter is an integer value in the range between 0 and 255). This is specified in the ECU Configuration Parameter Definition which is also an XML file. The relationship between the two file types is shown in figure 2.1.

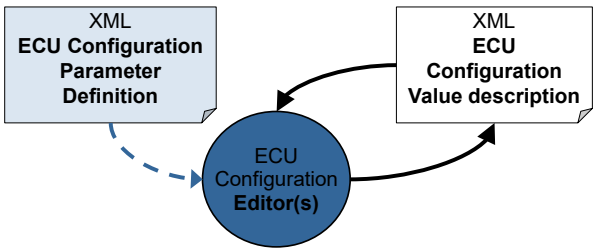

Figure 2.1: Parameter Definition and ECU Configuration Value files

For the ECU Configuration editors there are basically two possible approaches how to implement these definitions. Either the ECU Configuration Parameter Definition is read and interpreted directly from the XML file or the defined structures are hard-coded into the tool[1].

For the development of the ECU Configuration Parameter Definition and the ECU Configuration Values, a model-based approach has been chosen which already has been used during the development of other AUTOSAR template formats.

The main approach is to use a subset of UML to graphically model the desired entities and their relationships. Then, in a generation step, the actual XML formats are automatically generated out of the model.

[TPS_ECUC_02000] Modeling of ECU Configuration Value Meta-model and ECU Configuration Parameter Definition Meta-model ⌈ The modeling of the ECU Configuration Value Meta-model and ECU Configuration Parameter Definition Meta-model is done according to the Generic Structure Template [4].⌋

Please note that the Generic Structure Template [4] contains some fundamental infrastructure meta-classes and common patterns and provides details about:

- AUTOSAR Top level structure,
- Commonly used meta-classes and primitives
- Variant Handling
- Documentation

[TPS_ECUC_02001] Transformation of the ECU Configuration Value Meta-model and ECU Configuration Parameter Definition Meta-model to schema definitions ⌈ The transformation of the ECU Configuration Values Meta-model and ECU Configuration Parameter Definition Meta-model to schema definitions is done according to the XML Schema Production Rules [5].⌋

[1]The advantage of using the interpreter is that changes on the ECU Configuration Parameter Definition are directly available in the tool. But the hard-coded approach allows for more custom user support in the tool

Because of these transformation rules there is a given discrepancy between the UML model and the generated XML-Schema names. This also affects this document. The major descriptions will be based on the UML model notations (figures and tables), although the corresponding XML notation might be given for reference purposes.

In this section the application of the modeling approach for the ECU Configuration is described.

AUTOSAR uses the UML meta-model (M2-level) to describe the classes and objects that may be used in an AUTOSAR-compliant system. These meta-model elements may be used in an application model (M1-level) to describe the content of a real vehicle. ECU Configuration is a part of the AUTOSAR standard so the elements of `ECU Configuration Values` shall be described in the UML meta-model at M2-level. The (M2) meta-model has therefore been populated with UML descriptions from which ECU Configuration Parameter models may be built.

With M2 definitions in place, it is possible to create AUTOSAR-conforming models of real application ECU Configuration Parameters (an `ECU Configuration Parameter Definition Model`) at M1-level. Certain aspects of real application configurations are already defined: BSW Modules have standard interfaces and configuration requirements. These 'real' configuration parameters have therefore already been modeled at M1-level for each defined BSW Module. These are described in detail in the SWS documents.

XML has been chosen as the technology that will be used by AUTOSAR-compliant tools in order to define and share information during an AUTOSAR-compliant system development. It shall therefore be possible to transform the *UML* Configuration Parameter Definition Model (M1-level) into an *XML* Configuration Parameter Definition so that it may be used by ECU Configuration tools. This is the way that the tool gets a definition of exactly which ECU Configuration Parameters are available and how they may be configured. The XML Schema Production Rules [5] describes how the UML meta-model (M2-level) may be transformed into a schema that describes the format of XML to contain model elements.

This same formalization is also true for the `ECU Configuration Parameter Definition Meta-model` elements on M2-level: the XML Schema Production Rules dictate how `ECU Configuration Parameter Definition` elements will generate a schema to hold `ECU Configuration Parameter Model` (M1-level) elements in an XML `ECU Configuration Parameter Definition`, that can then be interpreted by ECU Configuration tools.

ECU Configuration editors allow a system designer to set ECU Configuration Parameter Values for their particular application. The actual values are then stored in an `ECU Configuration Values ARXML` that conforms to the template described in the UML.

An `ECU Configuration Values ARXML` is an XML file that conforms to an AUTOSAR schema called an ECU Configuration Value Template. The template in turn is an AUTOSAR standard defined by placing ECU Configuration Value Template

elements into the UML Meta-Model (M2-level) such that the schema (the ECU Configuration Value Template) can be generated (using the Formalization Guide rules).

There are three different parts involved in the development of the ECU Configuration: UML models, Schema and XML content files. The overview is shown in figure 2.2.

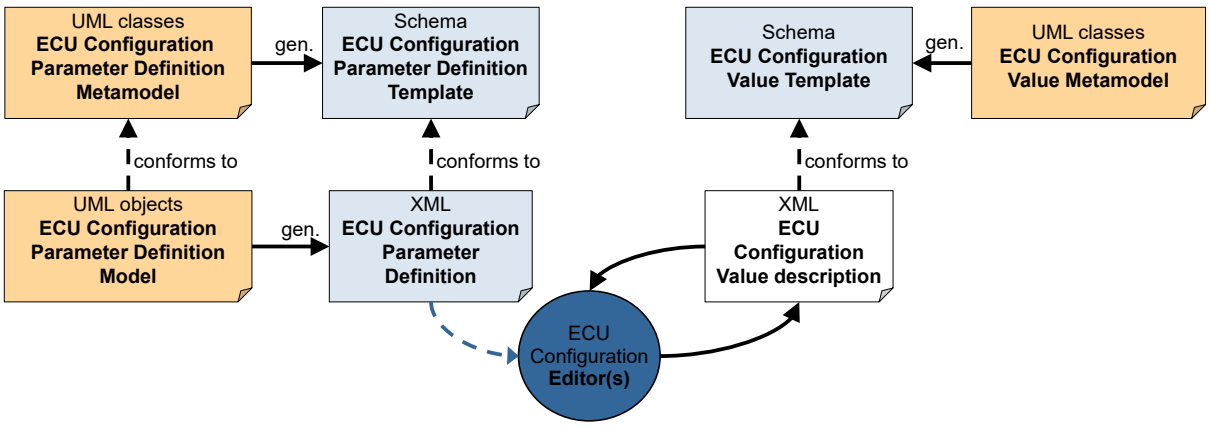

Figure 2.2: Relationship between UML models and XML files

The following section describes one way to define ECU Configuration Parameter Definitions. Other ways of defining and maintaining of ECU Configuration Parameter Definitions are also possible.

The ECU Configuration Parameter Definition Model (and UML object model) is used to specify the ECU Configuration Parameter Definition. This is done using object diagrams (this is the M1 level of meta-modeling) with special semantics defined in section 2.3. What kind of UML elements are allowed in the ECU Configuration Parameter Definition Model is defined in the ECU Configuration Parameter Definition Meta-model, which is conforming to the Generic Structure Template [4]. The definition is done using UML class diagrams (which is done on M2 level of meta-modeling).

Out of the ECU Configuration Parameter Definition Meta-model, a schema[2] is generated, and the generated ECU Configuration Parameter Definition XML file has to conform to this schema. Vendor-Specific ECU Configuration Parameter Definitions need to conform to this schema as well.

The ECU Configuration Values ARXML file needs to conform to the ECU Configuration Value Template (in the form of a schema), which itself is generated out of the ECU Configuration Value Metamodel, specified in UML class diagrams as well.

In the next section, the ECU Configuration Parameter Definition Meta-model and its application toward the ECU Configuration Parameter Definition Model is described.

In the following figures and tables, the names from the UML model are shown. In the generated XML-Schema, the names may differ based on the XML Schema Production Rules [5]. For instance, the attribute shortName will become SHORT-NAME in the XML-Schema.

[2]Whether a DTD or an XML-Schema is used is not relevant for this explanation and is left to the formalization strategy defined in [5].

2.3 ECU Configuration Parameter Definition Meta-model

As the name implies, the `ECU Configuration Parameter Definition` (sometimes abbreviated as ParamDef) defines the existence of ECU Configuration Parameters, their types and their attributes. The two major building blocks for the specification of ECU Configuration Parameter Definitions are containers and parameters/references. With the ability to establish relationships between containers and parameters and the means to specify references, the definition of parameters has enough power for the needs of the ECU Configuration. These and similar model elements, which form the `ECU Configuration Parameter Definition Meta-model`, are introduced in the following subsections.

2.3.1 ECU Configuration Parameter Definition top-level structure

The definition of each Software Module's[3] configuration has at the top level the structure shown in figure 2.3. For an overview of the complete ECU Configuration top level structure please refer to chapter 2.4.1.

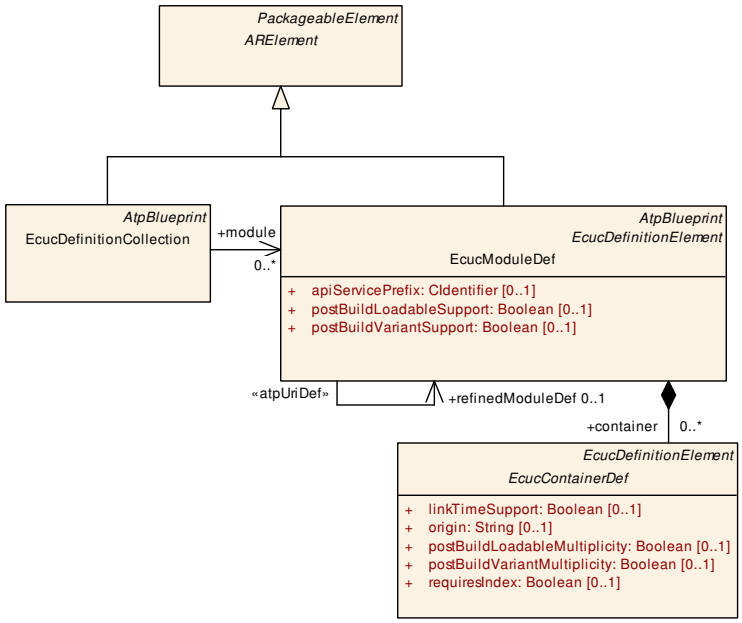

Figure 2.3: ECU Configuration Parameter Definition top-level structure

[TPS_ECUC_02002] Generic structure of all AUTOSAR templates ⌈ The generic structure of all AUTOSAR templates is described in detail in the AUTOSAR Generic Structure Template [4].⌋

[3] A Software Module might be Basic Software, RTE, Application Software Component or Complex Driver; see AUTOSAR Glossary [6]. The approach of Ecu configuration may be applied to non-standardized AUTOSAR Software modules (Application Software Component or Complex Driver) using the Vendor Specific Module Definition.

[TPS_ECUC_02130] Standardized Module Definition package structure ⌈ The Standardized Module Definition (StMD) as delivered by AUTOSAR [7] shall be provided inside the package structure */AUTOSAR/EcucDefs/*.⌋

[TPS_ECUC_06070] Sorting of ECU Configuration Parameter Definitions ⌈ ECU Configuration Parameter Definitions shall be sorted alphabetically.⌋

[TPS_ECUC_02003] `EcucDefinitionCollection` class ⌈ First ECU Configuration specific class is the `EcucDefinitionCollection` which inherits from `ARElement`. Through this inheritance the `EcucDefinitionCollection` can be part of an AUTOSAR `ARPackage` and thus part of an AUTOSAR description.⌋

[constr_4200] Existence of `EcucDefinitionCollection.module`

Imposition time: CP: `IT_EcucDef`

⌈`EcucDefinitionCollection.module` shall exist at least once.⌋

An `EcucDefinitionCollection` without any references to `EcucModuleDef` would not have any effect on the Ecu configuration.

[TPS_ECUC_02065] `EcucModuleDef` class ⌈ The ECU Configuration Parameter Definition of one module is called `EcucModuleDef` and inherits from `ARElement`.⌋

`ARElement` itself inherits from `PackageableElement`, `Identifiable` and `Referrable` which has two consequences: First, each `Referrable` has to have a machine readable `shortName`. Second, the `Identifiable` introduces the concept of a namespace for the contained `Identifiable` objects, so those objects need to have unique `shortName`s in the scope of that namespace. For additional information about the consequences of being a `Referrable` and `Identifiable` and the additional attributes please refer to the AUTOSAR Generic Structure Template [4].

[TPS_ECUC_02004] `EcucDefinitionCollection` collects all references to individual module configuration definitions ⌈ The use-case of the `EcucDefinitionCollection` class is to collect all references to individual module configuration definitions of the AUTOSAR ECU Configuration. Therefore the `EcucDefinitionCollection` specifies a reference relationship to the definition of several Software Modules in the `module` attribute.⌋

Please note that it is allowed to have several `EcucDefinitionCollection`s to collect the `EcucModuleDef`s based on various criteria e.g. modules from different vendors.

Class	**EcucDefinitionCollection**
Note	This represents the anchor point of an ECU Configuration Parameter Definition within the AUTOSAR templates structure. **Tags:** atp.recommendedPackage=EcucDefs This Class is only used by the AUTOSAR Classic Platform.

▽

△

Class	EcucDefinitionCollection			
Base	ARElement, ARObject, AtpBlueprint, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
module	EcucModuleDef	*	ref	References to the module definitions of individual software modules. **Tags:** atp.Splitkey=module

Table 2.1: EcucDefinitionCollection

2.3.1.1 Usage of the Admin Data

AdminData is an attribute of Identifiable [4] and can be used to set administrative information for an element (e.g. version information). Such administrative information can be set for the whole ECU Configuration Parameter Definition XML file and for each module definition.

[TPS_ECUC_06004] AdminData field in ECU Configuration Parameter Definition XML file ⌈ An AdminData field is required at the beginning of every ECU Configuration Parameter Definition XML file (regardless whether it is the StMD or the VSMD file) to allow the setting of AdminData for the whole XML File.⌋

Listing 2.1 shows how AdminData can be used for the whole ECU Configuration Parameter Definition file. For the files provided by AUTOSAR the AdminData shall be filled out with the AUTOSAR release information (Release and Revision number). For the files provided by Vendor the AdminData shall be filled out with the Vendor release information.

```
<ADMIN-DATA>
  <DOC-REVISIONS>
    <DOC-REVISION>
      <!--optional_file_revision-->
      <REVISION-LABEL>4.1.0</REVISION-LABEL>
      <!--AUTOSAR_or_VendorShortName-->
      <ISSUED-BY>AUTOSAR</ISSUED-BY>
      <!--optional_file_date-->
      <DATE>2014-10-31</DATE>
    </DOC-REVISION>
  </DOC-REVISIONS>
</ADMIN-DATA>
```

Listing 2.1: AdminData Example

[TPS_ECUC_06005] Usage of AdminData on EcucModuleDef is mandatory ⌈For each module definition, the revision of the StMD shall be provided. For the VSMD the AUTOSAR release version and the vendor's own version information shall be provided. The usage of AdminData on EcucModuleDef is mandatory.⌋

[TPS_ECUC_08053] AUTOSAR release version in VSMD ⌈In the VSMD the AUTOSAR release version shall be provided in following format:

- `DocRevision.revisionLabel` shall be set to the AUTOSAR release number.
- `DocRevision.issuedBy` shall be set to AUTOSAR.

⌋

Listing 2.2 shows that there are possibilities to specify several elements for the `AdminData`. The initial one would be provided by AUTOSAR, the additional one is the vendor's information which is based on the AUTOSAR one.

```
<ECUC-MODULE-DEF>
  <SHORT-NAME>Rte</SHORT-NAME>
  <ADMIN-DATA>
    <DOC-REVISIONS>
      <DOC-REVISION>
        <REVISION-LABEL>4.2.1</REVISION-LABEL>
        <ISSUED-BY>AUTOSAR</ISSUED-BY>
        <DATE>2014-10-31</DATE>
      </DOC-REVISION>
      <DOC-REVISION>
        <REVISION-LABEL>15.3.0</REVISION-LABEL>
        <!--predecessor -->
        <REVISION-LABEL-P-1>2.1.1</REVISION-LABEL-P-1>
        <ISSUED-BY>VendorX</ISSUED-BY>
        <DATE>2007-06-21T09:30:00+01:00</DATE>
      </DOC-REVISION>
    </DOC-REVISIONS>
  </ADMIN-DATA>
  <INTRODUCTION>
    <P>
      <L-1 L="EN">Configuration of the Rte (Runtime Environment) module.</L
          -1>
    </P>
  </INTRODUCTION>
  <LOWER-MULTIPLICITY>0</LOWER-MULTIPLICITY>
  <UPPER-MULTIPLICITY>1</UPPER-MULTIPLICITY>
  <CONTAINERS>
    <!-- ... -->
  </CONTAINERS>
</ECUC-MODULE-DEF>
```

Listing 2.2: `AdminData` with additional vendor information

2.3.1.2 Life Cycle definition

AUTOSAR provides support for life cycle handling, defined in the Generic Structure Template [4]. A standardized usage of this approach is defined in the Standardization Template [3].

For the definition of ECU Configuration Parameters there is support in the Meta-Model to annotate the life cycle state of each `EcucDefinitionElement`. For the annotation

the tag atp.Status can be used. In an Ecu Configuration Parameter Definition, this information is transported in a LifeCycleInfoSet, see Listing 2.3.

```
<LIFE-CYCLE-INFO-SET>
  <SHORT-NAME>AUTOSARParameterDefinition</SHORT-NAME>
  <LIFE-CYCLE-INFOS>
    <LIFE-CYCLE-INFO>
      <LC-OBJECT-REF DEST="ECUC-DEFINITION-ELEMENT">/AUTOSAR/EcucDefs/EcuC/
          EcucConfigSet/EcucPduCollection/Pdu/PduBufferAlignment</LC-OBJECT-
          REF>
      <LC-STATE-REF DEST="LIFE-CYCLE-STATE">/AUTOSAR/GenDef/
          LifeCycleStateDefinitionGroups/AutosarLifeCycleStates/draft</LC-
          STATE-REF>
    </LIFE-CYCLE-INFO>
  </LIFE-CYCLE-INFOS>
  <USED-LIFE-CYCLE-STATE-DEFINITION-GROUP-REF DEST="LIFE-CYCLE-STATE-
      DEFINITION-GROUP">/AUTOSAR/GenDef/LifeCycleStateDefinitionGroups/
      AutosarLifeCycleStates</USED-LIFE-CYCLE-STATE-DEFINITION-GROUP-REF>
</LIFE-CYCLE-INFO-SET>
```

Listing 2.3: `LifeCycle` Example

If a EcucParamConfContainerDef in the StMD has the atp.Status set to a value then the included aggregation of parameters, references and subContainers are allowed to have the atp.Status set according to [TPS_ECUC_06091].

There “1” means allowed and “0” means not allowed combination. Please note that the [TPS_ECUC_06091] is only valid for the StMD and not for the VSMD.

[TPS_ECUC_06091] Matrix of allowed status value combinations of EcucParamConfContainerDef and aggregations of EcucParameterDef/EcucAbstractReferenceDef/EcucContainerDef in the StMD ⌈

	Status of included aggregation			
Status of EcucParamConfContainerDef	draft	valid	obsolete	removed
draft	1	0	1	1
valid	1	1	1	1
obsolete	0	0	1	1
removed	0	0	0	1

⌋

If a EcucAbstractReferenceDef in the StMD has the atp.Status set to a value then the target of this reference is allowed to have the atp.Status set according to [TPS_ECUC_06092].

[TPS_ECUC_06092] Matrix of allowed status value combinations of referenced targets of a `EcucAbstractReferenceDef` ⌈

	Status of reference target			
Status of EcucAbstractReferenceDef	draft	valid	obsolete	removed
draft	1	1	0	0
valid	0	1	0	0
obsolete	1	1	1	0
removed	1	1	1	1

⌋

Please note that in the current StMD only the `atp.Status` values “valid” and “draft” are used. Additional handling for backwards compatibility can be done in the VSMD, for example keeping parameters in “obsolete” after they have been removed from the StMD. Therefore, there is no requirement for the StMD to set the parameters `LifeCycleState` to “obsolete” or “removed”, instead they are completely removed from the StMD.

The parameters in `CP MOD ECUConfigurationParmeters` follow the convention that if no `LifeCycleState` exists, the status is “valid”.

2.3.1.3 Documentation Support

AUTOSAR provides support for integrated and well-structured documentation. More details about the AUTOSAR Documentation Support concept can be found in the AUTOSAR Generic Structure Template [4].

The documentation can be specified within in the following levels:

- a single paragraph can be inserted in any `Identifiable` element using the `desc` element.
- a documentation block is available in any `Identifiable` element as `introduction`. This type of documentation is typically used to capture a short introduction about the role of an element or respectively how it is built.
- a standalone documentation structured into multiple chapters is also offered in AUTOSAR. It is provided as `Documentation` which is an `ARElement` of its own rights allowing for a reference to the documents context.

It is possible to split the container and parameter notes in the ECU Configuration Parameter Definition XML file into a `desc` and an `introduction` field. The `desc` field contains a brief description about the element and the `introduction` field contains the documentation about how the element is built and used.

In the past, this split was done in the StMD. Starting in R26-11, the complete note is inside the `introduction`; `desc` is no longer used by the StMD. Of course, a VSMD can still choose to split up the note between these two elements. In this case, the

content of the desc and introduction is read as one cohesive note. An example listing can be found in previous versions of the standard, in this document.

Listing 2.4 shows the usage of the Documentation element to describe elements like chapters, lists, tables and figures. For details on this description means please refer to the AUTOSAR Generic Structure Template [4].

```
<DOCUMENTATION>
  <SHORT-NAME>Adc_AddInfo</SHORT-NAME>
  <CONTEXTS>
    <DOCUMENTATION-CONTEXT>
      <SHORT-NAME>AUTOSAR_Adc</SHORT-NAME>
      <IDENTIFIABLE-REF DEST="ECUC-MODULE-DEF">/AUTOSAR/Adc</IDENTIFIABLE-
          REF>
    </DOCUMENTATION-CONTEXT>
  </CONTEXTS>
  <DOCUMENTATION-CONTENT>
    <CHAPTER>
      <SHORT-NAME>Introduction</SHORT-NAME>
      <P><L-1 L="EN">The ADC module initializes and controls the internal
          Analogue Digital Converter Unit(s) of the microcontroller. It
          provides services to start and stop a conversion respectively to
          enable and disable the trigger source for a conversion.</L-1></P>
      <P><L-1 L="EN">The consistency of the group channel results can be
          obtained with the following methods on the application side:</L-1>
          </P>
      <LIST>
        <ITEM>
          <P><L-1 L="EN">Using group notification mechanism</L-1></P>
        </ITEM>
        <ITEM>
          <P><L-1 L="EN">Polling via API function Adc_GetGroupStatus</L-1><
              /P>
        </ITEM>
      </LIST>
      <TABLE>
        <TGROUP COLS="2">
          <THEAD>
            <ROW>
              <ENTRY>
                <P><L-1 L="EN">column1</L-1></P>
              </ENTRY>
              <ENTRY>
                <P><L-1 L="EN">column2</L-1></P>
              </ENTRY>
            </ROW>
          </THEAD>
          <TBODY>
            <ROW>
              <ENTRY>
                <P><L-1 L="EN">element11</L-1></P>
              </ENTRY>
              <ENTRY>
                <P><L-1 L="EN">element12</L-1></P>
              </ENTRY>
            </ROW>
```

```
            <ROW>
              <ENTRY>
                <P><L-1 L="EN">element21</L-1></P>
              </ENTRY>
              <ENTRY>
                <P><L-1 L="EN">element22</L-1></P>
              </ENTRY>
            </ROW>
          </TBODY>
        </TGROUP>
      </TABLE>
    </CHAPTER>
  </DOCUMENTATION-CONTENT>
</DOCUMENTATION>
```

Listing 2.4: `Documentation` Example

2.3.2 ECU Configuration Module Definition

[TPS_ECUC_02005] `EcucModuleDef` class ⌈ The class `EcucModuleDef` is defining the ECU Configuration Parameters of one Software Module[4]. It is inheriting from `ARElement`, so each individual `EcucModuleDef` needs to have a unique name within its enclosing `ARPackage`.⌋

[TPS_ECUC_02059] Number of instances of a BSW module in the `ECU Configuration Values` ⌈ The `EcucModuleDef` is using the `EcucDefinitionElement` attributes to specify how many instances of that specific module are allowed in the `ECU Configuration Values` (see [TPS_ECUC_02008] for more details).⌋

Class	**EcucModuleDef**			
Note	Used as the top-level element for configuration definition for Software Modules, including BSW and RTE as well as ECU Infrastructure. **Tags:** atp.recommendedPackage=EcucDefs This Class is only used by the AUTOSAR Classic Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpDefinition*, *CollectableElement*, *EcucDefinitionElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
apiServicePrefix	CIdentifier	0..1	attr	For modules where several instances of the VSMD can be defined the apiServicePrefix defines the API namespace of the derived instances, e.g. Cdd, Xfrm (ComXf, SomeIpXf, E2EXf).

▽

[4] A Software Module is not restricted to the BSW Modules but also includes the RTE, Application Software Components and generic ECU Configuration.

△

Class	**EcucModuleDef**			
container	EcucContainerDef	*	aggr	Aggregates the top-level container definitions of this specific module definition. **Tags:** atp.Splitkey=container, container.shortName xml.sequenceOffset=11
postBuild Loadable Support	Boolean	0..1	attr	Indicates if the module supports changing the configuration by loading a new configuration binary at postBuild time. TRUE means yes, FALSE means no.
postBuildVariant Support	Boolean	0..1	attr	Indicates if a module supports different post-build variants (previously known as post-build selectable configuration sets). TRUE means yes, FALSE means no.
refinedModule Def	EcucModuleDef	0..1	ref	Optional reference from the Vendor Specific Module Definition to the Standardized Module Definition it refines. In case this EcucModuleDef has the category STANDARDIZED_MODULE_DEFINITION this reference shall not be provided. In case this EcucModuleDef has the category VENDOR_SPECIFIC_MODULE_ DEFINITION this reference is mandatory. **Stereotypes:** atpUriDef **Tags:** atp.Splitkey=refinedModuleDef
supported ConfigVariant	EcucConfiguration VariantEnum	*	attr	Specifies which ConfigurationVariants are supported by this software module. This attribute is optional if the Ecuc ModuleDef has the category STANDARDIZED_ MODULE_DEFINITION. If the category attribute of the EcucModuleDef is set to VENDOR_SPECIFIC_ MODULE_DEFINITION then this attribute is mandatory. **Tags:** atp.Status=obsolete

Table 2.2: EcucModuleDef

[TPS_ECUC_02094] `EcucModuleDef` is able to aggregate container definitions
⌈ The `EcucModuleDef` aggregates container definitions (`EcucModuleDef`) with the role name `container` which may hold other container definitions, parameter definitions and reference definitions.⌋

[constr_4201] Existence of `EcucModuleDef.container`

Imposition time: CP: `IT_EcucDef`

⌈ The reference `EcucModuleDef.container` shall exist at least once.⌋

An `EcucModuleDef` without any `container` would not have any effect on the Ecu configuration.

[TPS_ECUC_08012] Module support for post-build variants ⌈The `postBuildVariantSupport` attribute of the `EcucModuleDef` specifies if this `EcucModuleDef` supports different post-build selectable configuration sets (post-build variants). `true` means yes, `false` means no.⌋

[constr_9439] Existence of attribute `EcucModuleDef.postBuildVariantSupport`

Imposition time: CP: `IT_EcucDef`

⌈For each `EcucModuleDef`, the attribute `postBuildVariantSupport` shall exist.⌋

[TPS_ECUC_06094] Module support for post-build loadable ⌈The postBuildLoadableSupport attribute of the EcucModuleDef specifies if this EcucModuleDef supports the update of the configuration at post-build time by loading it into the persistent storage of the Ecu. postBuildLoadableSupport = true means yes, postBuildLoadableSupport = false means no.⌋

[constr_9440] Existence of attribute EcucModuleDef.postBuildLoadableSupport

Imposition time: CP: IT_EcucDef

⌈For each EcucModuleDef, the attribute postBuildLoadableSupport shall exist.⌋

[constr_5507] Value of EcucContainerDef.postBuildVariantMultiplicity if postBuildVariantSupport is set to false

Imposition time: CP: IT_EcucDef

⌈If postBuildVariantSupport is set to false, every EcucContainerDef in this EcucModuleDef with upperMultiplicity greater than lowerMultiplicity shall have its postBuildVariantMultiplicity attribute set to false.⌋

[constr_9441] Value of EcucContainerDef.postBuildLoadableMultiplicity if postBuildLoadableSupport is set to false

Imposition time: CP: IT_EcucDef

⌈If postBuildLoadableSupport is set to false, every EcucContainerDef in this EcucModuleDef with upperMultiplicity greater than lowerMultiplicity shall have its postBuildLoadableMultiplicity attribute set to false.⌋

[constr_5509] Value of postBuildVariantMultiplicity if postBuildVariantSupport is set to false

Imposition time: CP: IT_EcucDef

⌈If postBuildVariantSupport is set to false, every EcucCommonAttributes in this EcucModuleDef with upperMultiplicity greater than lowerMultiplicity shall have its postBuildVariantMultiplicity attribute set to false.⌋

[constr_9442] Value of postBuildLoadableMultiplicity if postBuildLoadableSupport is set to false

Imposition time: CP: IT_EcucDef

⌈If postBuildLoadableSupport is set to false, every EcucCommonAttributes in this EcucModuleDef with upperMultiplicity greater than lowerMultiplicity shall have its postBuildLoadableMultiplicity attribute set to false.⌋

[constr_5510] Value of postBuildVariantValue if postBuildVariantSupport is set to false

Imposition time: CP: IT_EcucDef

⌈If postBuildVariantSupport is set to false, every EcucCommonAttributes in this EcucModuleDef shall have its postBuildVariantValue attribute set to false.⌋

[constr_9443] Value of postBuildLoadableValue if postBuildLoadableSupport is set to false

Imposition time: CP: IT_EcucDef

⌈If postBuildLoadableSupport is set to false, every EcucCommonAttributes in this EcucModuleDef shall have its postBuildLoadableValue attribute set to false.⌋

[TPS_ECUC_02095] VSMD refines the StMD ⌈ The reference refinedModuleDef from an EcucModuleDef with the category *VENDOR_SPECIFIC_MODULE_DEFINITION* to an EcucModuleDef with the category *STANDARDIZED_MODULE_DEFINITION* specifies that the source EcucModuleDef is the *Vendor Specific Module Definition* which refines the referenced Standardized EcucModuleDef.⌋

[TPS_ECUC_06076] Use cases where the reference refinedModuleDef is mandatory ⌈The refinedModuleDef reference is mandatory if the EcucModuleDef with the category *VENDOR_SPECIFIC_MODULE_DEFINITION* actually refines the EcucModuleDef with the category *STANDARDIZED_MODULE_DEFINITION* (e.g. Vendor Specific Module Definition of Com BSW module refines Standardized Module Definition of Com BSW module).⌋

[TPS_ECUC_06077] Use cases where the reference refinedModuleDef is optional ⌈The refinedModuleDef reference is not necessary if the EcucModuleDef with the category *VENDOR_SPECIFIC_MODULE_DEFINITION* does not actually refines any EcucModuleDefs with the category *STANDARDIZED_MODULE_DEFINITION* (e.g. Vendor Specific Module Definition of CDD which does not contribute to the ComStack configuration).⌋

[TPS_ECUC_06044] No refinedModuleDef reference in the StMD ⌈ The reference refinedModuleDef from an EcucModuleDef with the category *STANDARDIZED_MODULE_DEFINITION* shall not be used.⌋

[TPS_ECUC_06043] EcucModuleDef class categories ⌈

category	*Meaning*
STANDARDIZED_MODULE_DEFINITION	The EcucModuleDef class is used to describe the Standardized Module Definition (StMD)

▽

△

category	*Meaning*
VENDOR_SPECIFIC_MODULE_DEFINITION	The EcucModuleDef class is used to describe Vendor Specific Module Definition

The category attribute shall be used to clearly distinguish between the different roles of the EcucModuleDef class.

⌋

[constr_3022] EcucModuleDef category restriction

Imposition time: CP: IT_EcucDef

⌈The category definition shall be restricted to exactly the two defined ones:

- VENDOR_SPECIFIC_MODULE_DEFINITION
- STANDARDIZED_MODULE_DEFINITION

⌋

In figure 2.4 an example of the top-level structure is provided and in the Listing 2.5 the corresponding ECU Configuration Parameter Definition XML file extract is shown. In the example XML also the overall XML structure of AUTOSAR descriptions is shown. The corresponding ECU Configuration Values ARXML file extract is shown in Listing 2.26.

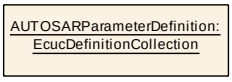

Figure 2.4: ECU Configuration Parameter Definition example

```
<AR-PACKAGE>
  <SHORT-NAME>EcucDefs</SHORT-NAME>
  <ELEMENTS>
    <ECUC-DEFINITION-COLLECTION>
      <SHORT-NAME>AUTOSARParameterDefinition</SHORT-NAME>
      <MODULE-REFS>
        <MODULE-REF DEST="ECUC-MODULE-DEF">/AUTOSAR/EcucDefs/Rte</MODULE-
            REF>
        <!-- Further references to module definitions -->
      </MODULE-REFS>
    </ECUC-DEFINITION-COLLECTION>
    <ECUC-MODULE-DEF>
      <SHORT-NAME>Rte</SHORT-NAME>
      <DESC>
        <L-2 L="EN">Configuration Parameter Definition of the RTE</L-2>
      </DESC>
      <LOWER-MULTIPLICITY>0</LOWER-MULTIPLICITY>
      <UPPER-MULTIPLICITY>1</UPPER-MULTIPLICITY>
      <POST-BUILD-LOADABLE-SUPPORT>false</POST-BUILD-LOADABLE-SUPPORT>
      <POST-BUILD-VARIANT-SUPPORT>false</POST-BUILD-VARIANT-SUPPORT>
      <CONTAINERS>
        <!-- ... -->
      </CONTAINERS>
    </ECUC-MODULE-DEF>
```

```
  </ELEMENTS>
</AR-PACKAGE>
```

Listing 2.5: `EcucModuleDef` top-level structure

In the next sections the structure of containers, individual parameters and references is introduced.

2.3.3 Container Definition

There are two specializations of a container definition. The abstract class `EcucContainerDef` is used to gather the common features (see figure 2.5).

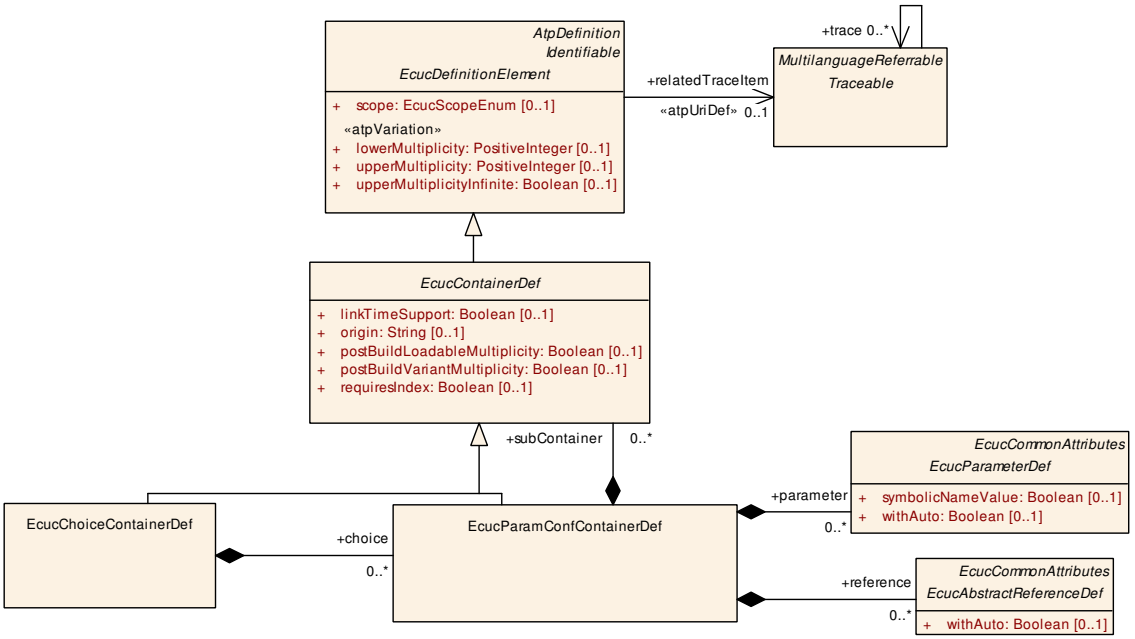

Figure 2.5: Class diagram for parameter container definition

Class	***EcucContainerDef*** (abstract)			
Note	Base class used to gather common attributes of configuration container definitions. This Class is only used by the AUTOSAR Classic Platform.			
Base	*ARObject*, *AtpDefinition*, *EcucDefinitionElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	EcucChoiceContainerDef, EcucParamConfContainerDef			
Aggregated by	EcucDestinationUriPolicy.container, EcucModuleDef.container, EcucParamConfContainerDef.subContainer			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
destinationUri	EcucDestinationUriDef	*	ref	Several destinationUris can be defined for an EcucContainerDef. With such destinationUris an EcucContainerDef is applicable for several EcucUriReferenceDefs. **Stereotypes:** atpUriDef **Tags:** atp.Splitkey=destinationUri

△

Class	***EcucContainerDef*** (abstract)			
linkTime Support	Boolean	0..1	attr	Indicates whether the container multiplicity can be changed at LinkTime. Please note that Link-Time is only supported in the Vendor Specific Module Definition (VSMD).
multiplicity ConfigClass	EcucMultiplicity ConfigurationClass	*	aggr	Specifies which MultiplicityConfigurationClass this container is available for which ConfigurationVariant. This aggregation is optional if the surrounding EcucModuleDef has the Category STANDARDIZED_MODULE_ DEFINITION. If the category attribute of the EcucModule Def is set to VENDOR_SPECIFIC_MODULE_ DEFINITION and if the upperMultiplicity is greater than the lowerMultiplicity then this aggregation is mandatory. **Tags:** atp.Splitkey=multiplicityConfigClass atp.Status=obsolete xml.namePlural=MULTIPLICITY-CONFIG-CLASSES
origin	String	0..1	attr	This attribute specifies whether this configuration container is an AUTOSAR standardized container or whether it is vendor-specific.
postBuild Loadable Multiplicity	Boolean	0..1	attr	Indicates whether the container multiplicity can be changed at post-build loadable time by updating the configuration binary. TRUE means yes, FALSE means no.
postBuildVariant Multiplicity	Boolean	0..1	attr	Indicates if a container may have different number of instances in different post-build variants (previously known as post-build selectable configuration sets). TRUE means yes, FALSE means no.
requiresIndex	Boolean	0..1	attr	Used to define whether the value element for this definition shall be provided with an index.

Table 2.3: EcucContainerDef

[TPS_ECUC_02044] Number of instances of a `EcucContainerDef` in the `ECU Configuration Values` ⌈ Each `EcucContainerDef` also has the features of `EcucDefinitionElement` which enables to specify for each `EcucContainerDef` how often it is allowed to occur in the `ECU Configuration Values` later on (see [TPS_ECUC_02008] for more details).⌋

[TPS_ECUC_06095] Semantics of `postBuildLoadableMultiplicity` attribute ⌈The `postBuildLoadableMultiplicity` attribute specifies whether the number of instances of this `EcucContainerDef` may change at post-build loadable time. `postBuildLoadableMultiplicity` = `true` means yes, `postBuildLoadable-Multiplicity` = `false` means no.⌋

[TPS_ECUC_06096] Latest configuration time to change the number of instances of `EcucContainerDef`s according to the StMD ⌈Specification of the latest time when the number of instances of an `EcucContainerDef` may change if the `Ecuc-ContainerDef` is aggregated by an `EcucModuleDef` with the `category` *STANDARDIZED_MODULE_DEFINITION*:

- `PreCompile` - if the attributes `EcucContainerDef.postBuild-LoadableMultiplicity` and `EcucContainerDef.postBuildVariant-Multiplicity` are set to false.

- PostBuildLoadable - if the attribute EcucContainerDef.postBuildLoadableMultiplicity is set to true.
- PostBuildSelectable - if the attribute EcucContainerDef.postBuildVariantMultiplicity is set to true.

⌋

[TPS_ECUC_06097] Latest configuration time to change the number of instances of EcucContainerDefs according to the VSMD ⌈Specification of the latest time when the number of instances of an EcucContainerDef may change if the EcucContainerDef is aggregated by an EcucModuleDef with the category *VENDOR_SPECIFIC_MODULE_DEFINITION*:

- PreCompile - if the attributes EcucContainerDef.postBuildLoadableMultiplicity, EcucContainerDef.postBuildVariantMultiplicity, and EcucContainerDef.linkTimeSupport are set to false.
- LinkTime - if the attribute EcucContainerDef.linkTimeSupport is set to true.
- PostBuildLoadable - if the attribute EcucContainerDef.postBuildLoadableMultiplicity is set to true.
- PostBuildSelectable - if the attribute EcucContainerDef.postBuildVariantMultiplicity is set to true.

⌋

[constr_9444] Mutual exclusivity of linkTimeSupport and postBuildLoadableMultiplicity

Imposition time: CP: IT_EcucDef

⌈The attributes EcucContainerDef.linkTimeSupport and EcucContainerDef.postBuildLoadableMultiplicity are mutually exclusive, meaning zero or just one of them shall be used.⌋

[constr_9445] EcucContainerDef.linkTimeSupport not applicable in StMD

Imposition time: CP: IT_EcucDef

⌈The EcucContainerDef.linkTimeSupport attribute shall not be used by an EcucContainerDef that is aggregated by an EcucModuleDef with the category *STANDARDIZED_MODULE_DEFINITION*.⌋

Link-Time configuration is supported as a feature in the AUTOSAR standard. All EcucContainerDefs that are declared as PostBuild capable in the StMD can be refined to LinkTime in the VSMD.

[constr_9446] Existence of the postBuildLoadableMultiplicity attribute

Imposition time: CP: IT_EcucDef

⌈For each EcucContainerDef for which the upperMultiplicity is greater than the lowerMultiplicity, the attribute postBuildLoadableMultiplicity shall exist.⌋

[constr_9447] Removing an instance of the EcucContainerDef at post-build time

Imposition time: CP: IT_EcucDef

⌈Only instances of EcucContainerDefs with postBuildLoadable-Multiplicity or postBuildVariantMultiplicity set to true which are not referenced or are exclusively referenced by EcucAbstractReferenceDefs with postBuildLoadableValue or postBuildVariantValue set to true and have been introduced at post-build time (not part of the initial configuration before post-build updates) can be removed at post-build time.⌋

[TPS_ECUC_06120] Usage of postBuildLoadableMultiplicity attribute is independent of its aggregated subContainers ⌈An EcucContainerDef may have the attribute postBuildLoadableMultiplicity set to true even if one or more of its aggregated EcucContainerDefs in the role subContainer have the attribute postBuildLoadableMultiplicity set to false.⌋

If a container "A" has the attribute postBuildLoadableMultiplicity set to true and its sub-container "B" has the attribute postBuildLoadableMultiplicity set to false, it is not possible to add a new instance "b2" of sub-container "B" to the existing container instance "a1" of "A" in post-build loadable time. However, it is allowed to add a new instance "a2" of the container "A" together with a new instance "b2" of its sub-container "B".

[TPS_ECUC_08013] Different number of EcucContainerDef instances in different post-build selectable variants ⌈The postBuildVariantMultiplicity attribute of the EcucContainerDef specifies if a different number of instances of this EcucContainerDef may exist in different post-build selectable configuration sets (post-build variants). true means yes, false means no.⌋

[constr_5506] Existence of postBuildVariantMultiplicity attribute ⌈For each EcucContainerDef for which the upperMultiplicity is greater than the lowerMultiplicity, the attribute postBuildVariantMultiplicity shall exist.⌋

[TPS_ECUC_08014] Usage of postBuildVariantMultiplicity attribute is independent of aggregated subContainers ⌈An EcucContainerDef may have the attribute postBuildVariantMultiplicity set to true even if one or more of its aggregated EcucContainerDefs in the role subContainer have the attribute postBuildVariantMultiplicity set to false.⌋

If container "A" has postBuildVariantMultiplicity attribute set to true and its sub-container "B" set to false, it is not possible to have a different number of instances of "B" in the same instance of "A" in different post-build variants. However it is allowed to have a different number of instances of container "A" where new instances may have arbitrary number of instances of container "B".

[TPS_ECUC_06118] EcucModuleDef that relies on EcucContainerDefs with postBuildVariantMultiplicity set to true of another EcucModuleDef ⌈If one EcucModuleDef relies on the EcucContainerDefs with postBuildVariant-Multiplicity set to true of another EcucModuleDef, the number of instances of these EcucContainerDefs can only differ in different post-build selectable configuration sets (post-build variants) if the using EcucModuleDef has the postBuild-VariantSupport set to true as well.⌋

[TPS_ECUC_06119] EcucModuleDef that relies on EcucContainerDefs with postBuildLoadableMultiplicity set to true of another EcucModuleDef ⌈If one EcucModuleDef relies on the EcucContainerDefs with postBuild-LoadableMultiplicity set to true of another EcucModuleDef, the number of instances of these EcucContainerDefs can only be changed at post-build time if the using EcucModuleDef has the postBuildLoadableSupport set to true as well.⌋

[TPS_ECUC_02007] EcucParamConfContainerDef class ⌈ A EcucParamConf-ContainerDef is the main container class definition and can contain other containers, configuration parameters and references.⌋

Class	**EcucParamConfContainerDef**			
Note	Used to define configuration containers that can hierarchically contain other containers and/or parameter definitions. This Class is only used by the AUTOSAR Classic Platform.			
Base	*ARObject*, *AtpDefinition*, *EcucContainerDef*, *EcucDefinitionElement*, *Identifiable*, *Multilanguage Referrable*, *Referrable*			
Aggregated by	EcucChoiceContainerDef.choice, EcucDestinationUriPolicy.container, EcucModuleDef.container, Ecuc ParamConfContainerDef.subContainer			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
parameter	EcucParameterDef	*	aggr	The parameters defined within the EcucParamConf ContainerDef. **Tags:** atp.Splitkey=parameter, parameter.shortName
reference	EcucAbstractReference Def	*	aggr	The references defined within the EcucParamConf ContainerDef. **Tags:** atp.Splitkey=reference, reference.shortName
subContainer	EcucContainerDef	*	aggr	The containers defined within the EcucParamConf ContainerDef. **Tags:** atp.Splitkey=subContainer, subContainer.short Name

Table 2.4: EcucParamConfContainerDef

One example of a EcucContainerDef and its embedding in the ECU Configuration Parameter Definition is shown in figure 2.6. One EcucModuleDef Rte is specified being part of the EcucDefinitionCollection. Two containers of type EcucParam-ConfContainerDef are specified as part of the module definition.

When specifying the containment relationship between the EcucModuleDef and containers the role name container is used. When specifying the containment relationship between two containers an aggregation with the role name subContainer at the contained container is used.

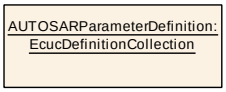

Figure 2.6: Example of an object diagram for container definition

In the XML snippet in Listing 2.6 only the relevant part from figure 2.6 is shown, not including the EcucDefinitionCollection[5]. The corresponding ECU Configuration Values ARXML file extract is shown in Listing 2.30.

```
<ECUC-MODULE-DEF>
  <SHORT-NAME>Rte</SHORT-NAME>
  <LOWER-MULTIPLICITY>0</LOWER-MULTIPLICITY>
  <UPPER-MULTIPLICITY>1</UPPER-MULTIPLICITY>
  <POST-BUILD-LOADABLE-SUPPORT>false</POST-BUILD-LOADABLE-SUPPORT>
  <POST-BUILD-VARIANT-SUPPORT>false</POST-BUILD-VARIANT-SUPPORT>
  <CONTAINERS>
    <ECUC-PARAM-CONF-CONTAINER-DEF>
      <SHORT-NAME>RteGeneration</SHORT-NAME>
      <LOWER-MULTIPLICITY>1</LOWER-MULTIPLICITY>
      <UPPER-MULTIPLICITY>1</UPPER-MULTIPLICITY>
    </ECUC-PARAM-CONF-CONTAINER-DEF>
    <ECUC-PARAM-CONF-CONTAINER-DEF>
      <SHORT-NAME>SwComponentInstance</SHORT-NAME>
      <LOWER-MULTIPLICITY>1</LOWER-MULTIPLICITY>
      <UPPER-MULTIPLICITY-INFINITE>true</UPPER-MULTIPLICITY-INFINITE>
      <POST-BUILD-LOADABLE-MULTIPLICITY>false</POST-BUILD-LOADABLE-
         MULTIPLICITY>
      <POST-BUILD-VARIANT-MULTIPLICITY>false</POST-BUILD-VARIANT-
         MULTIPLICITY>
    </ECUC-PARAM-CONF-CONTAINER-DEF>
  </CONTAINERS>
</ECUC-MODULE-DEF>
```

Listing 2.6: EcucContainerDef Example

2.3.3.1 Choice Container Definition

[TPS_ECUC_02011] EcucChoiceContainerDef class ⌈ The EcucChoiceContainerDef can be used to specify that certain containers might occur exclusively in the ECU Configuration Values. In the ECU Configuration Parameter Definition, the potential containers are specified as part of the EcucChoiceContainerDef, and the constraint is that in the actual ECU Configuration Values only some of those specified containers will actually be present.⌋

[5]Note that in the figures of ECU Configuration Parameter Definition modeled in UML the infinite upper multiplicity is shown as *upperMultiplicity* = * resulting in <*UPPER-MULTIPLICITY-INFINITE*>*true*</*UPPER-MULTIPLICITY-INFINITE*>

Class	**EcucChoiceContainerDef**			
Note	Used to define configuration containers that provide a choice between several EcucParamConfContainer Def. But in the actual ECU Configuration Values only one instance from the choice list will be present. This Class is only used by the AUTOSAR Classic Platform.			
Base	*ARObject*, *AtpDefinition*, *EcucContainerDef*, *EcucDefinitionElement*, *Identifiable*, *Multilanguage Referrable*, *Referrable*			
Aggregated by	EcucDestinationUriPolicy.container, EcucModuleDef.container, EcucParamConfContainerDef.sub Container			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
choice	EcucParamConf ContainerDef	*	aggr	The choices available in a EcucChoiceContainerDef. **Tags:** atp.Splitkey=choice, choice.shortName

Table 2.5: EcucChoiceContainerDef

[TPS_ECUC_02067] Multiplicity of the *to be chosen* containers ⌈ The multiplicity of the *to be chosen* containers shall always be `0..1`, indicating that each time a choice is performed you can only choose exactly one of these *to be chosen* containers at a time.⌋

[TPS_ECUC_02012] Allowed choice of available *to be chosen* containers in the `ECU Configuration Values` ⌈ Each time a choice can be performed, the user is free to choose one of the available *to be chosen* containers. The `upperMultiplicity` of the `EcucChoiceContainerDef` specifies how many instances on the values side shall be allowed.⌋

An example of the usage of a `EcucChoiceContainerDef` is shown in figure 2.7 and the XML definition is shown in Listing 2.7. The corresponding `ECU Configuration Values ARXML` is shown in Listing 2.31.

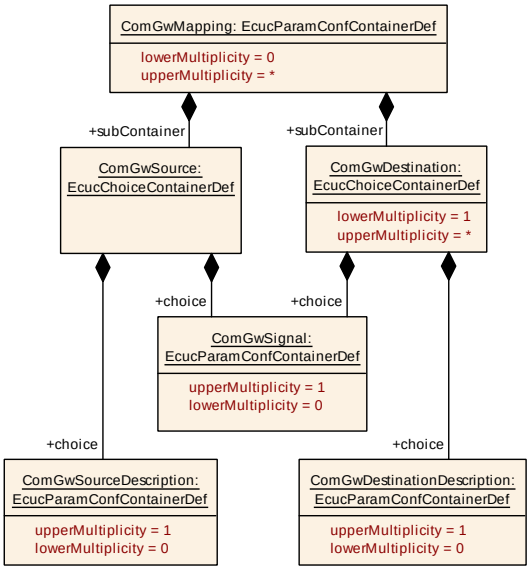

Figure 2.7: Example of an object diagram for two choice container definitions

The example shows two use-cases of `EcucChoiceContainerDef` with different multiplicities of the `EcucChoiceContainerDef`.

The EcucChoiceContainerDef ComGwSource is defined to be able to hold one of the two given containers later in the ECU Configuration Values. Since the upperMultiplicity of ComGwSource = 1, there can only be one choice taken.

The EcucChoiceContainerDef ComGwDestination is defined to be able to hold one of the two given containers later in the ECU Configuration Values. Since the upperMultiplicity of ComGwDestination = *, several choices can be taken.

```
<ECUC-MODULE-DEF>
  <SHORT-NAME>Com</SHORT-NAME>
  <LOWER-MULTIPLICITY>1</LOWER-MULTIPLICITY>
  <UPPER-MULTIPLICITY>1</UPPER-MULTIPLICITY>
  <POST-BUILD-LOADABLE-SUPPORT>true</POST-BUILD-LOADABLE-SUPPORT>
  <POST-BUILD-VARIANT-SUPPORT>true</POST-BUILD-VARIANT-SUPPORT>
  <CONTAINERS>
    <ECUC-PARAM-CONF-CONTAINER-DEF>
      <SHORT-NAME>ComGwMapping</SHORT-NAME>
      <LOWER-MULTIPLICITY>0</LOWER-MULTIPLICITY>
      <UPPER-MULTIPLICITY-INFINITE>true</UPPER-MULTIPLICITY-INFINITE>
      <POST-BUILD-LOADABLE-MULTIPLICITY>true</POST-BUILD-LOADABLE-
          MULTIPLICITY>
      <POST-BUILD-VARIANT-MULTIPLICITY>true</POST-BUILD-VARIANT-
          MULTIPLICITY>
      <SUB-CONTAINERS>
        <ECUC-CHOICE-CONTAINER-DEF>
          <SHORT-NAME>ComGwSource</SHORT-NAME>
          <LOWER-MULTIPLICITY>1</LOWER-MULTIPLICITY>
          <UPPER-MULTIPLICITY>1</UPPER-MULTIPLICITY>
          <CHOICES>
            <ECUC-PARAM-CONF-CONTAINER-DEF>
              <SHORT-NAME>ComGwSignal</SHORT-NAME>
              <!-- ... -->
            </ECUC-PARAM-CONF-CONTAINER-DEF>
            <ECUC-PARAM-CONF-CONTAINER-DEF>
              <SHORT-NAME>ComGwSourceDescription</SHORT-NAME>
              <!-- ... -->
            </ECUC-PARAM-CONF-CONTAINER-DEF>
          </CHOICES>
        </ECUC-CHOICE-CONTAINER-DEF>
        <ECUC-CHOICE-CONTAINER-DEF>
          <SHORT-NAME>ComGwDestination</SHORT-NAME>
          <LOWER-MULTIPLICITY>1</LOWER-MULTIPLICITY>
          <UPPER-MULTIPLICITY-INFINITE>true</UPPER-MULTIPLICITY-INFINITE>
          <POST-BUILD-LOADABLE-MULTIPLICITY>true</POST-BUILD-LOADABLE-
              MULTIPLICITY>
          <POST-BUILD-VARIANT-MULTIPLICITY>true</POST-BUILD-VARIANT-
              MULTIPLICITY>
          <CHOICES>
            <ECUC-PARAM-CONF-CONTAINER-DEF>
              <SHORT-NAME>ComGwSignal</SHORT-NAME>
              <!-- ... -->
            </ECUC-PARAM-CONF-CONTAINER-DEF>
            <ECUC-PARAM-CONF-CONTAINER-DEF>
              <SHORT-NAME>ComGwDestinationDescription</SHORT-NAME>
              <!-- ... -->
            </ECUC-PARAM-CONF-CONTAINER-DEF>
```

```
          </CHOICES>
        </ECUC-CHOICE-CONTAINER-DEF>
      </SUB-CONTAINERS>
    </ECUC-PARAM-CONF-CONTAINER-DEF>
  </CONTAINERS>
</ECUC-MODULE-DEF>
```

Listing 2.7: `EcucChoiceContainerDef` Example

The containers from the example, which the choice is from, will of course have to be specified in more detail in an actual definition file.

2.3.4 Common Configuration Elements

Configuration Containers, Parameters and References have some common attributes which are described in this section.

2.3.4.1 Variant Handling

Variant Handling has been introduced to AUTOSAR in a generic way. The major specification can be found in the AUTOSAR Generic Structure Template [4]. Every element which is subject to variability shall have the stereotype `<<atpVariation>>` set.

Variant Handling is used in both areas of ECU Configuration, the `ECU Configuration Parameter Definition` and `ECU Configuration Values`. In this specification the semantics of variant handling are specified at the actual location where they occur individually.

2.3.4.2 Configuration Multiplicity

[TPS_ECUC_02008] Number of occurrences of containers, parameters and references in the `ECU Configuration Values` ⌈ To be able to specify how often a specific configuration element (container, parameter or reference) may occur in the `ECU Configuration Values`, the class `EcucDefinitionElement` is introduced. With the two attributes `lowerMultiplicity` and `upperMultiplicity` the minimum and maximum occurrence of the configuration element is specified.⌋

[TPS_ECUC_06016] Countably infinite number of containers, parameters and references in the `ECU Configuration Values` ⌈To express a countable infinite number of occurrences of this element, the `upperMultiplicityInfinite` element shall exist and shall be set to `true`. [6]⌋

[6]Note that in the figures of `ECU Configuration Parameter Definition` modeled in UML, the infinite upper multiplicity is shown as *upperMultiplicity* = *

[constr_5325] Existence of `upperMultiplicityInfinite` and `upperMultiplicity` is mutually exclusive

Imposition time: CP: `IT_EcucDef`

⌈The existence of the elements `upperMultiplicityInfinite` and `upperMultiplicity` shall be mutually exclusive.⌋

[TPS_ECUC_02110] Variable lower and upper multiplicity in ECU Configuration Parameter definition ⌈ The attributes `lowerMultiplicity`, `upperMultiplicity` and `upperMultiplicityInfinite` are subject to variant handling. The values can be computed using the variant handling mechanism.⌋

In this specification the literals `n` and `m` are used to represent some natural number in order to allow the definition of relations between the `lowerMultiplicity` and the `upperMultiplicity`.

[TPS_ECUC_02009] Expression of optionality of containers, parameters and references in the Ecuc Parameter Definition UML model ⌈If there is no multiplicity defined for an element in the Ecuc Parameter Definition UML model ([8] - AUTOSAR_MMOD_MetaModel), then the element will be generated as mandatory in the Ecu Parameter Definition ARXML file ([7] - Specification of ECU Configuration Parameters):

- `lowerMultiplicity` = 1
- `upperMultiplicity` = 1.

To express an optional element in the Ecuc Parameter Definition model ([8] - AUTOSAR_MMOD_MetaModel) the `lowerMultiplicity` has to be set explicitly to '`0`'.⌋

As Ecuc Parameter definition UML model figures can be quite large and consume a lot of graphical space, some space can be saved by omitting the `lowerMultiplicity` and `upperMultiplicity` respectively `upperMultiplicityInfinite`. In the generated Ecu Parameter Definition ARXML file these multiplicities are defined as mandatory.

Configuration Parameter and Reference definitions with an `upperMultiplicity` > `1` have to be considered with care, since it is not possible to reference to individual parameters. So such multiple occurrences of a parameter in the `ECU Configuration Values` will just be mere collections, it is neither guaranteed that the order will be preserved nor that individual elements do have a special semantics.

[TPS_ECUC_02010] Multiplicity attributes in ECU Configuration Parameter Model diagrams ⌈ In the specification object diagrams (ECU Configuration Parameter Model) the multiplicity attributes may be omitted if both values are equal to the default value of '`1`'. Otherwise both attributes are shown.⌋

Class	***EcucDefinitionElement*** (abstract)			
Note	Common class used to express the commonalities of configuration parameters, references and containers. If not stated otherwise the default multiplicity is exactly one mandatory occurrence of the specified element. This Class is only used by the AUTOSAR Classic Platform.			
Base	*ARObject*, *AtpDefinition*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	*EcucCommonAttributes*, *EcucContainerDef*, EcucModuleDef			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
ecucCond	EcucCondition Specification	0..1	aggr	If it evaluates to true the Ecu Parameter definition shall be processed as specified. Otherwise the parameter definition shall be ignored. **Tags:** atp.Splitkey=ecucCond xml.sequenceOffset=100
ecucValidation Cond	EcucValidation Condition	*	aggr	Collection of validation conditions which all need to evaluate to true in order to indicate a valid validation condition of the EcucDefinitionElement. **Tags:** atp.Splitkey=ecucValidationCond, ecucValidation Cond.shortName
lowerMultiplicity	PositiveInteger	0..1	attr	The lower multiplicity of the specified element. 0: optional 1: at least one occurrence n: at least n occurrences **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=codeGenerationTime xml.sequenceOffset=110
relatedTrace Item	Traceable	0..1	ref	This contains a sloppy reference to the Autosar compatible identifier of the element (EcucId). **Stereotypes:** atpUriDef **Tags:** atp.Splitkey=relatedTraceItem xml.sequenceOffset=-10
scope	EcucScopeEnum	0..1	attr	Specifies the scope of this configuration element. **Tags:** xml.sequenceOffset=150
upperMultiplicity	PositiveInteger	0..1	attr	The upper multiplicity of the specified element. 0: no occurrence (used for VSMD) 1: at most one occurrence m: at most m occurrences If upperMultiplicity is set than upperMultiplicityInfinite shall not be used. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=codeGenerationTime xml.sequenceOffset=120
upperMultiplicity Infinite	Boolean	0..1	attr	To express an infinite number of occurrences of this element this attribute has to be set to true. If upperMultiplicityInfinite is set than upperMultiplicity shall not be used. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=codeGenerationTime xml.sequenceOffset=130

Table 2.6: EcucDefinitionElement

Enumeration	**EcucScopeEnum**
Note	Possible scope settings for a configuration element. This Enumeration is only used by the AUTOSAR Classic Platform.
Aggregated by	*EcucDefinitionElement*.scope
Literal	***Description***
ECU	An element may be shared with other modules. **Tags:** atp.EnumerationLiteralIndex=0
local	An element is only be applicable for the module it is defined in. **Tags:** atp.EnumerationLiteralIndex=1

Table 2.7: EcucScopeEnum

The reference `EcucDefinitionElement.relatedTraceItem` is used to provide the Specification ID of the respective `EcucDefinitionElement` in the StMD. Please note that the same Specification ID can occur several times in the StMD because the same `EcucDefinitionElement` can be part of several `EcucParamConfContainerDef`s.

`EcucDefinitionElement.relatedTraceItem` can be used in the VSMD but the value shall not conflict with the AUTOSAR defined namespace.

[constr_3200] Restriction on values of `EcucDefinitionElement.relatedTraceItem` in the VSMD

Imposition time: CP: `IT_EcucDef`

⌈The value of `EcucDefinitionElement.relatedTraceItem` in the VSMD shall never start with 'ECUC_'.⌋

The values of `EcucDefinitionElement.relatedTraceItem` starting with 'ECUC_' are reserved for AUTOSAR standardization.

For examples please refer to figure 2.6 and Listing 2.6

2.3.4.3 Common Configuration Attributes

Several attributes are available on both, parameters and references. These common attributes are shown in figure 2.8.

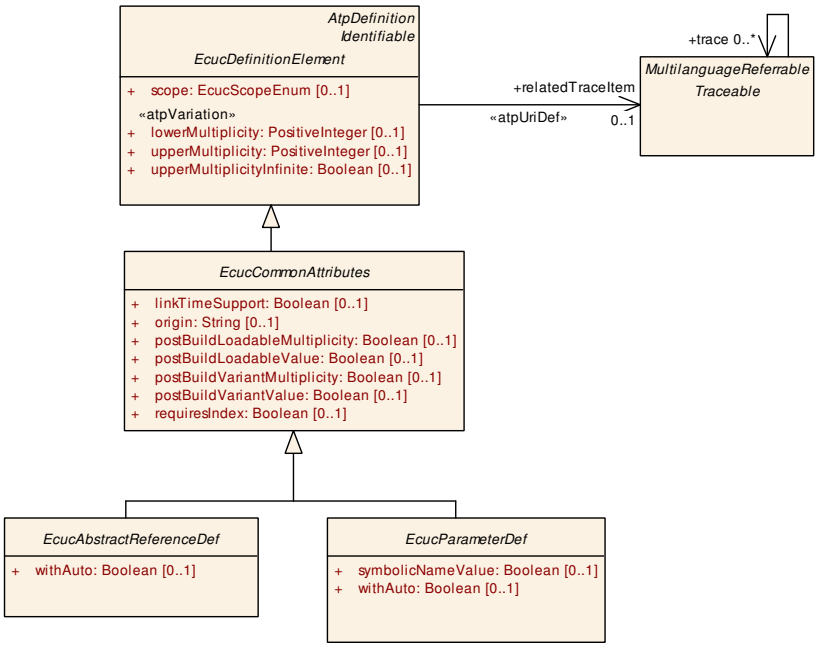

Figure 2.8: Common Attributes for parameters and references

[constr_3570] `EcucDefinitionElement.lowerMultiplicity` always required

Imposition time: CP: `IT_EcucDef`

⌈The attribute `EcucDefinitionElement.lowerMultiplicity` shall always be defined.⌋

[constr_5342] `EcucDefinitionElement.upperMultiplicity` or `EcucDefinitionElement.upperMultiplicityInfinite` required

Imposition time: CP: `IT_EcucDef`

⌈Exactly one of the attributes `EcucDefinitionElement.upperMultiplicity` or `EcucDefinitionElement.upperMultiplicityInfinite` shall be defined.⌋

[constr_3571] `EcucCommonAttributes.origin` always required

Imposition time: CP: `IT_EcucDef`

⌈The attribute `EcucCommonAttributes.origin` shall always be defined.⌋

[constr_3572] `EcucParameterDef.symbolicNameValue` always required

Imposition time: CP: `IT_EcucDef`

⌈The attribute `EcucParameterDef.symbolicNameValue` shall always be defined.⌋

Class	***EcucCommonAttributes*** (abstract)
Note	Attributes used by Configuration Parameters as well as References. This Class is only used by the AUTOSAR Classic Platform.
Base	*ARObject*, *AtpDefinition*, *EcucDefinitionElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*

△

Class	***EcucCommonAttributes*** (abstract)			
Subclasses	*EcucAbstractReferenceDef*, *EcucParameterDef*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
linkTime Support	Boolean	0..1	attr	Indicates whether the parameter or reference can be changed at LinkTime. Please note that Link-Time is only supported in the Vendor Specific Module Definition (VSMD).
multiplicity ConfigClass	EcucMultiplicity ConfigurationClass	*	aggr	Specifies in which MultiplicityConfigurationClass this parameter or reference is available in a particular ConfigurationVariant. This aggregation is optional if the surrounding EcucModuleDef has the Category STANDARDIZED_MODULE_DEFINITION. If the category attribute of the EcucModuleDef is set to VENDOR_SPECIFIC_MODULE_DEFINITION, then this aggregation is mandatory. **Tags:** atp.Splitkey=multiplicityConfigClass atp.Status=obsolete xml.namePlural=MULTIPLICITY-CONFIG-CLASSES
origin	String	0..1	attr	String specifying if this configuration parameter is an AUTOSAR standardized configuration parameter or if the parameter is hardware- or vendor-specific.
postBuild Loadable Multiplicity	Boolean	0..1	attr	Indicates whether the multiplicity of the parameter or reference can be changed at post-build loadable time by updating the configuration binary. TRUE means yes, FALSE means no.
postBuild LoadableValue	Boolean	0..1	attr	Indicates whether the parameter or reference value may be loaded to the memory at post-build time . TRUE means yes, FALSE means no.
postBuildVariant Multiplicity	Boolean	0..1	attr	Indicates if a parameter or a reference may have different number of instances in different post-build variants (previously known as post-build selectable configuration sets). TRUE means yes, FALSE means no.
postBuildVariant Value	Boolean	0..1	attr	Indicates if a parameter or a reference may have different value in different post-build variants (previously known as post-build selectable configuration sets). TRUE means yes, FALSE means no.
requiresIndex	Boolean	0..1	attr	Used to define whether the value element for this definition shall be provided with an index.
valueConfig Class	EcucValueConfiguration Class	*	aggr	Specifies in which ValueConfigurationClass this parameter or reference is available in a particular ConfigurationVariant. This aggregation is optional if the surrounding EcucModuleDef has the Category STANDARDIZED_MODULE_DEFINITION. If the category attribute of the EcucModuleDef is set to VENDOR_SPECIFIC_MODULE_DEFINITION, then this aggregation is mandatory. **Tags:** atp.Splitkey=valueConfigClass atp.Status=obsolete xml.namePlural=VALUE-CONFIG-CLASSES

Table 2.8: EcucCommonAttributes

2.3.4.3.1 Parameter Origin

[constr_5365] Origin information in parameter and reference definitions

Imposition time: CP: IT_EcucDef

⌈Each instance of the subclass of EcucCommonAttributes or EcucContainerDef shall provide a value for the origin attribute and this attribute shall be either:

- 'AUTOSAR_ECUC' - in case that the parameter definition is standardized by AUTOSAR
- vendor specific value - in case that the parameter definition is vendor specific. For vendor specific origins no rules are defined by AUTOSAR and the vendor is free to choose the value (e.g. 'VendorXYZ_v1.3').

⌋

```
<ECUC-INTEGER-PARAM-DEF>
  <SHORT-NAME>ClockRate</SHORT-NAME>
  <ORIGIN>AUTOSAR_ECUC</ORIGIN>
</ECUC-INTEGER-PARAM-DEF>
<ECUC-BOOLEAN-PARAM-DEF>
  <SHORT-NAME>VendorExtensionEnabled</SHORT-NAME>
  <ORIGIN>VendorXYZ_v1.3</ORIGIN>
</ECUC-BOOLEAN-PARAM-DEF>
```

Listing 2.8: origin Example

In Listing 2.8 two parameters are defined, one which belongs to the AUTOSAR standard and one which is introduced by the module vendor in a specific version of his own ECU Configuration tools.

2.3.4.3.2 Value and Multiplicity Configuration Classes

[TPS_ECUC_06098] Latest configuration time to change the number of instances of parameters or references according to the StMD ⌈Specification of the latest time when the number of instances of the parameter or reference may change if the EcucCommonAttributes is aggregated via an EcucContainerDef by an EcucModuleDef with the category *STANDARDIZED_MODULE_DEFINITION*:

- PreCompile - if the attributes postBuildLoadableMultiplicity and postBuildVariantMultiplicity are set to false.
- PostBuildLoadable - if the attribute postBuildLoadableMultiplicity is set to true.
- PostBuildSelectable - if the attribute postBuildVariantMultiplicity is set to true.

⌋

[TPS_ECUC_06099] Latest configuration time to change the value of parameters or references according to the StMD ⌈Specification of the latest time when the parameter or reference value may change if the EcucCommonAttributes is aggregated via an EcucContainerDef by an EcucModuleDef with the category *STANDARDIZED_MODULE_DEFINITION*:

- PreCompile - if the attributes postBuildLoadableValue and postBuildVariantValue are set to false.
- PostBuildLoadable - if the attribute postBuildLoadableValue is set to true.
- PostBuildSelectable - if the attribute postBuildVariantValue is set to true.

⌋

[constr_9450] EcucCommonAttributes.linkTimeSupport not applicable in StMD

Imposition time: CP: IT_EcucDef

⌈The EcucCommonAttributes.linkTimeSupport attributes shall not be used in EcucCommonAttributes that is aggregated by an EcucModuleDef with the category *STANDARDIZED_MODULE_DEFINITION*.⌋

[constr_9451] Existence of attribute EcucCommonAttributes.linkTimeSupport

Imposition time: CP: IT_EcucDef

⌈For each EcucModuleDef with the category *VENDOR_SPECIFIC_MODULE_DEFINITION*, the attribute linkTimeSupport shall exist.⌋

Link-Time configuration is supported as a feature in the AUTOSAR standard. All parameters and references that are declared as PostBuild capable in the StMD can be refined to LinkTime in the VSMD.

[TPS_ECUC_06100] Latest configuration time to change the number of instances of parameters or references according to the VSMD ⌈Specification of the latest time when the number of instances of the parameter or reference may change if the EcucCommonAttributes is aggregated via EcucContainerDef by an EcucModuleDef with the category *VENDOR_SPECIFIC_MODULE_DEFINITION*:

- PreCompile - if the attributes postBuildLoadableMultiplicity, postBuildVariantMultiplicity and linkTimeSupport are set to false.
- LinkTime - if the attribute linkTimeSupport is set to true.
- PostBuildLoadable - if the attribute postBuildLoadableMultiplicity is set to true.

- PostBuildSelectable - if the attribute postBuildVariantMultiplicity is set to true.

⌋

[constr_9452] Mutual exclusivity of linkTimeSupport and postBuildLoadableMultiplicity

Imposition time: CP: IT_EcucDef

⌈The attributes EcucCommonAttributes.linkTimeSupport and EcucCommonAttributes.postBuildLoadableMultiplicity are mutually exclusive, meaning zero or just one of them shall be used.⌋

[constr_9453] Mutual exclusivity of linkTimeSupport and postBuildLoadableValue

Imposition time: CP: IT_EcucDef

⌈The attributes EcucCommonAttributes.linkTimeSupport and EcucCommonAttributes.postBuildLoadableValue are mutually exclusive, meaning zero or just one of them shall be used.⌋

[TPS_ECUC_06101] Latest configuration time to change the value of parameters or references according to the VSMD ⌈Specification of the latest time when the parameter or reference value may change if the EcucCommonAttributes is aggregated via EcucContainerDef by an EcucModuleDef with the category *VENDOR_SPECIFIC_MODULE_DEFINITION*:

- PreCompile - if the attributes postBuildLoadableValue, postBuildVariantValue, and linkTimeSupport are set to false.
- LinkTime - if the attribute linkTimeSupport is set to true.
- PostBuildLoadable - if the attribute postBuildLoadableValue is set to true.
- PostBuildSelectable - if the attribute postBuildVariantValue is set to true.

⌋

The element PublishedInformation is used to specify the fact that certain information is fixed even before the pre-compile stage.

The supported configuration variants of the module are described in section 2.3.2.

2.3.4.3.3 Value and Multiplicity Variant Values in Post-Build Selectable Variants

[TPS_ECUC_08016] Different values of EcucCommonAttributes instances in post-build selectable variants ⌈The postBuildVariantValue attribute of Ecuc-

CommonAttributes specifies if a different value of this EcucCommonAttributes may exist in different post-build variants. true means yes, false means no.⌋

[TPS_ECUC_08015] Different number of EcucCommonAttributess instances in post-build selectable variants ⌈The postBuildVariantMultiplicity attribute of EcucCommonAttributes specifies if a different number of instances of this EcucCommonAttributes may exist in different post-build selectable variants. true means yes, false means no.⌋

[TPS_ECUC_01059] EcucModuleDef that relies on EcucCommonAttributes with postBuildVariantValue set to true of another EcucModuleDef ⌈If one EcucModuleDef relies on the EcucCommonAttributes (parameters and references) with postBuildVariantValue set to true of another EcucModuleDef, the values of these EcucCommonAttributes can only differ in different post-build variants if the implementation of the using EcucModuleDef supports post-build variations.⌋

[TPS_ECUC_01060] EcucModuleDef that relies on EcucCommonAttributes with postBuildVariantMultiplicity set to true of another EcucModuleDef ⌈If one EcucModuleDef relies on the EcucCommonAttributes (parameters and references) with postBuildVariantMultiplicity set to true of another EcucModuleDef, the number of instances of these EcucCommonAttributes can only differ in different post-build variants if the implementation of the using EcucModuleDef supports post-build variations.⌋

[constr_9454] Existence of attribute EcucCommonAttributes.postBuildVariantValue

Imposition time: CP: IT_EcucDef

⌈For each EcucCommonAttributes, the attribute postBuildVariantValue shall exist.⌋

[constr_9455] Existence of attribute EcucCommonAttributes.postBuildVariantMultiplicity

Imposition time: CP: IT_EcucDef

⌈For each EcucCommonAttributes for which the upperMultiplicity is greater than the lowerMultiplicity, the attribute postBuildVariantMultiplicity shall exist.⌋

2.3.4.3.4 Value and Multiplicity Values in Post-Build Loadable Variants

[TPS_ECUC_06102] Semantics of EcucCommonAttributes.postBuildLoadableMultiplicity attribute ⌈The EcucCommonAttributes.postBuildLoadableMultiplicity attribute specifies whether the number of instances of this EcucParameterDef or EcucAbstractReferenceDef may change at post-build

loadable time. postBuildLoadableMultiplicity = true means yes, postBuildLoadableMultiplicity = false means no.⌋

[TPS_ECUC_06103] Semantics of EcucCommonAttributes.postBuildLoadableValue attribute ⌈The EcucCommonAttributes.postBuildLoadableValue attribute specifies whether the value of instances of this EcucParameterDef or EcucAbstractReferenceDef may change at post-build loadable time. postBuildLoadableMultiplicity = true means yes, postBuildLoadableMultiplicity = false means no.⌋

[constr_9456] Existence of attribute EcucCommonAttributes.postBuildLoadableValue

Imposition time: CP: IT_EcucDef

⌈For each EcucCommonAttributes, the attribute postBuildLoadableValue shall exist.⌋

[constr_9457] Existence of attribute EcucCommonAttributes.postBuildLoadableMultiplicity

Imposition time: CP: IT_EcucDef

⌈For each EcucCommonAttributes for which the upperMultiplicity is greater than the lowerMultiplicity, the attribute postBuildLoadableMultiplicity shall exist.⌋

[constr_9458] EcucModuleDef that relies on EcucCommonAttributes with postBuildLoadableValue set to true of another EcucModuleDef

Imposition time: CP: IT_EcucDef

⌈If one EcucModuleDef relies on the EcucCommonAttributes (parameters and references) with postBuildLoadableValue set to true of another EcucModuleDef, the values of these EcucCommonAttributes can only differ in different post-build loadable configurations if the implementation of the using EcucModuleDef has the EcucModuleDef.postBuildLoadableSupport set to true.⌋

[constr_9459] EcucModuleDef that relies on EcucCommonAttributes with postBuildLoadableMultiplicity set to true of another EcucModuleDef

Imposition time: CP: IT_EcucDef

⌈If one EcucModuleDef relies on the EcucCommonAttributes (parameters and references) with postBuildLoadableMultiplicity set to true of another EcucModuleDef, the number of instances of these EcucCommonAttributes can only differ in different post-build loadable configurations if the implementation of the using EcucModuleDef has the EcucModuleDef.postBuildLoadableSupport set to true.⌋

2.3.5 Parameter Definition

[TPS_ECUC_02013] Definition of parameters within a EcucParamConfContainerDef ⌈ Parameters are defined within a EcucParamConfContainerDef using an aggregation with the role name parameter at the parameter side.⌋

[TPS_ECUC_02014] Parameter types ⌈ The possible parameter types are specified using one of the specialized classes derived from EcucParameterDef. The EcucParameterDef does inherit from Identifiable, EcucCommonAttributes and EcucDefinitionElement.⌋

The available parameter types are shown in figure 2.9.

[TPS_ECUC_06104] EcucModuleDef that relies on EcucCommonAttributes with postBuildLoadableValue = true of another EcucModuleDef ⌈If one EcucModuleDef relies on the EcucCommonAttributes (parameters and references) with postBuildLoadableValue = true, the values of these EcucCommonAttributes can only be changed at post-build loadable time if the corresponding EcucModuleConfigurationValues of the using EcucModuleDef has the postBuildLoadableSupport set to true, respectively.⌋

[TPS_ECUC_06105] EcucModuleDef that relies on EcucCommonAttributes with postBuildVariantValue = true of another EcucModuleDef ⌈If one EcucModuleDef relies on the EcucCommonAttributes (parameters and references) with postBuildVariantValue = true, the values of these EcucCommonAttributes can only be changed at post-build selectable time if the corresponding EcucModuleConfigurationValues of the using EcucModuleDef has the postBuildVariantSupport set to true, respectively.⌋

[TPS_ECUC_06106] EcucModuleDef that relies on EcucCommonAttributes with postBuildLoadableMultiplicity = true of another EcucModuleDef ⌈If one EcucModuleDef relies on the EcucCommonAttributes (parameters and references) with postBuildLoadableMultiplicity = true, the number of instances of these EcucCommonAttributes can only be changed at post-build loadable time if the corresponding EcucModuleConfigurationValues of the using EcucModuleDef has the postBuildLoadableSupport set to true, respectively.⌋

[TPS_ECUC_06107] EcucModuleDef that relies on EcucCommonAttributes with postBuildVariantMultiplicity = true of another EcucModuleDef ⌈If one EcucModuleDef relies on the EcucCommonAttributes (parameters and references) with postBuildVariantMultiplicity = true, the number of instances of these EcucCommonAttributes can only be changed at post-build selectable time if the corresponding EcucModuleConfigurationValues of the using EcucModuleDef has the postBuildVariantSupport set to true, respectively.⌋

[TPS_ECUC_06108] EcucModuleDef that relies on EcucCommonAttributes with linkTimeSupport = true of another EcucModuleDef ⌈If one EcucModule-

Def relies on the EcucCommonAttributes (parameters and references) with linkTimeSupport = true, the values of these EcucCommonAttributes can only be changed at link time if the corresponding EcucModuleConfigurationValues of the using EcucModuleDef has the postBuildVariantSupport set to true, respectively.⌋

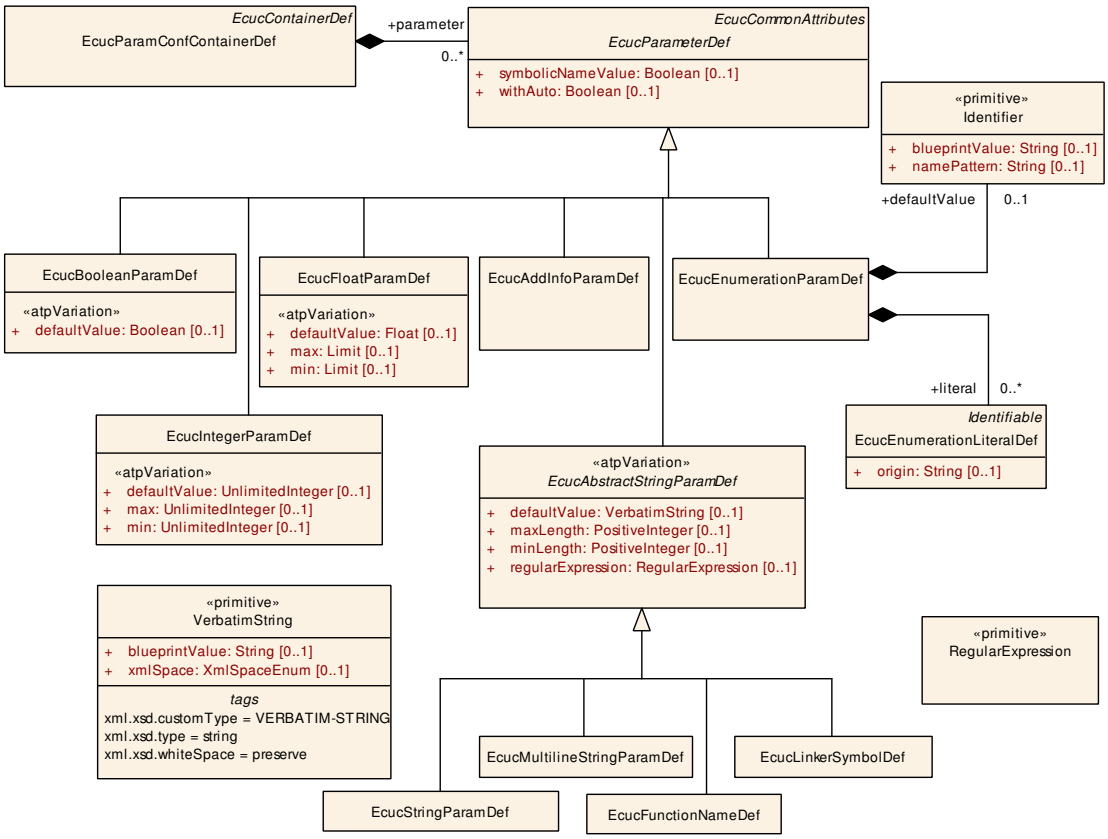

Figure 2.9: Class diagram for parameter definition

[constr_3575] EcucEnumerationLiteralDef.origin always required

Imposition time: CP: IT_EcucDef

⌈The attribute EcucEnumerationLiteralDef.origin shall always be defined.⌋

Class	***EcucParameterDef*** (abstract)
Note	Abstract class used to define the similarities of all ECU Configuration Parameter types defined as subclasses. This Class is only used by the AUTOSAR Classic Platform.
Base	*ARObject*, *AtpDefinition*, *EcucCommonAttributes*, *EcucDefinitionElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*
Subclasses	*EcucAbstractStringParamDef*, EcucAddInfoParamDef, EcucBooleanParamDef, EcucEnumerationParamDef, EcucFloatParamDef, EcucIntegerParamDef
Aggregated by	EcucParamConfContainerDef.parameter

△

Class	*EcucParameterDef* (abstract)			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
derivation	EcucDerivation Specification	0..1	aggr	A derivation of a Configuration Parameter value can be specified by an informal Calculation Formula or by a formal language that can be used to specify the computational rules. **Tags:** atp.Splitkey=derivation
symbolicName Value	Boolean	0..1	attr	Specifies that this parameter's value is used, together with the aggregating container, to derive a symbolic name definition. See chapter "Representation of Symbolic Names" in Ecuc specification for more details.
withAuto	Boolean	0..1	attr	Specifies whether it shall be allowed on the value side to specify this parameter value as "AUTO". If withAuto is "true" it shall be possible to set the "isAuto Value" attribute of the respective parameter to "true". This means that the actual value will not be considered during ECU Configuration but will be (re-)calculated by the code generator and stored in the value attribute afterwards. These implicit updated values might require a re-generation of other modules which reference these values. If withAuto is "false" it shall not be possible to set the "is AutoValue" attribute of the respective parameter to "true". If withAuto is not present the default is "false".

Table 2.9: EcucParameterDef

The use-case for the attribute `symbolicNameValue` is described in section 2.3.6.5.

The use-case for the attribute `withAuto` is described in section 3.4.1.

In the next sections the different parameter types will be described in detail. The examples for the individual parameters are taken from figure 2.10.

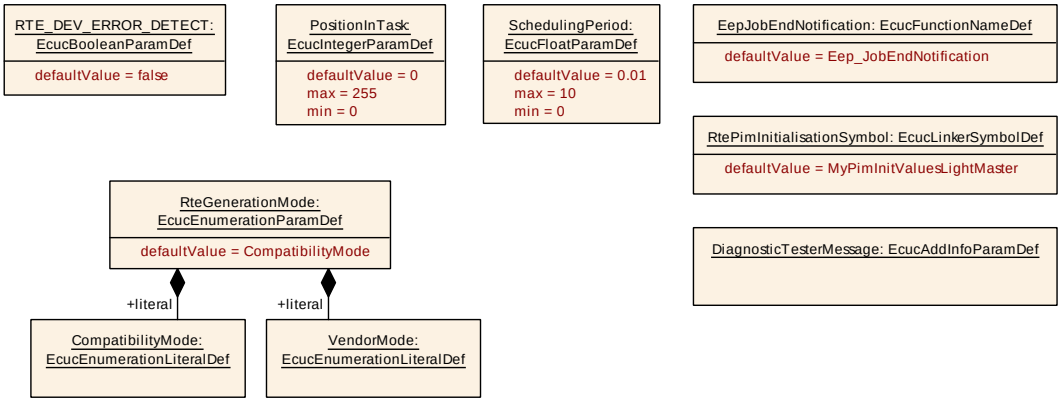

Figure 2.10: Example of parameter definitions using different types

2.3.5.1 Boolean Type

[TPS_ECUC_02026] `EcucBooleanParamDef` properties ⌈ With the `EcucBooleanParamDef` parameter a 'true' or 'false' parameter can be specified. The only ad-

ditional attribute is the defaultValue which may be specified while defining the parameter.⌋

[TPS_ECUC_02127] Possible values for EcucBooleanParamDef parameters ⌈ The alternative representation of 'true' and 'false' are '1' and '0' which allows the usage of a numerical representation of the value in order to be computed in the variant handling.⌋

This parameter can also be used for other 'boolean'-type configuration parameters with the semantic of:

- ON / OFF
- ENABLE / DISABLE
- 1 / 0

even if the ECU Configuration Values are restricted as described in [TPS_ECUC_02127].

Please note that the representation of an boolean parameter value or an attribute which supports ≪atpVariation≫ as true / false already requires the processing of the BooleanLiteral true /false by the formula processor.

On the ECU Configuration Values side, boolean parameter values are represented as EcucNumericalParamValues (see chapter 2.4.4.2). The attribute "value" in the EcucNumericalParamValue supports ≪atpVariation≫ and therefore the BooleanLiteral true /false is supported by the formula language as well. Please note that true evaluates to 1 and false to 0 (see [4] for more details).

[TPS_ECUC_02111] Variable default value in EcucBooleanParamDef ⌈ The attribute defaultValue of EcucBooleanParamDef is subject to variant handling. The value can be computed using the variant handling mechanism.⌋

Class	**EcucBooleanParamDef**			
Note	Configuration parameter type for Boolean. Allowed values are true and false. This Class is only used by the AUTOSAR Classic Platform.			
Base	*ARObject*, *AtpDefinition*, *EcucCommonAttributes*, *EcucDefinitionElement*, *EcucParameterDef*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	EcucParamConfContainerDef.parameter			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
defaultValue	Boolean	0..1	attr	Default value of the boolean configuration parameter. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=codeGenerationTime

Table 2.10: EcucBooleanParamDef

Listing 2.9 shows the ECUC Parameter definition XML file. The corresponding ECU Configuration Values ARXML file extract is shown in Listing 2.35.

```
<ECUC-BOOLEAN-PARAM-DEF>
  <SHORT-NAME>BswMDevErrorDetect</SHORT-NAME>
```

```
  <DEFAULT-VALUE>false</DEFAULT-VALUE>
</ECUC-BOOLEAN-PARAM-DEF>
```

Listing 2.9: `EcucBooleanParamDef` Example

2.3.5.2 Integer Type

[TPS_ECUC_02027] `EcucIntegerParamDef` properties ⌈ With the `EcucIntegerParamDef` parameter a signed/unsigned whole number can be specified. With the additional attributes `min` and `max`, the range of this parameters values in the `ECU Configuration Values` can be limited[7]. Also the `defaultValue` can be specified.⌋

[TPS_ECUC_02114] Variable default value in `EcucIntegerParamDef` ⌈ The attribute `defaultValue` of `EcucIntegerParamDef` is subject to variant handling. The value can be computed using the variant handling mechanism.⌋

[TPS_ECUC_02116] Variable min, max values in `EcucIntegerParamDef` ⌈ The attributes `min` and `max` of `EcucIntegerParamDef` are subject to variant handling. The values can be computed using the variant handling mechanism.⌋

The value range of the `EcucIntegerParamDef` has two use-cases, signed and unsigned, which both have to fit in a 64-bit number space.

[TPS_ECUC_02072] Signed `EcucIntegerParamDef` value range ⌈If a signed value is represented the `min` value can be down to `-9223372036854775808` and the `max` value can be up to `9223372036854775807`.⌋

[TPS_ECUC_02073] Unsigned `EcucIntegerParamDef` value range ⌈ If an unsigned value is represented the `min` value can be down to `0` and the `max` value can be up to `18446744073709551615` (in hex `0xFFFFFFFFFFFFFFFF`).⌋

[TPS_ECUC_06032] Min and max values in `EcucIntegerParamDef` ⌈ The `max` value shall be equal or bigger than the `min` value and the `min` value shall be equal or less than the `max` value.⌋

[TPS_ECUC_02074] +/- sign in the `EcucNumericalParamValue` is optional ⌈If the optional +/- sign in the `EcucNumericalParamValue` is omitted, "+" is assumed.⌋

[TPS_ECUC_03040] The value of an `EcucNumericalParamValue` shall be unambiguously an integer value ⌈The value of an `EcucNumericalParamValue` shall be specified such that it is unambiguously an integer value. In particular the result of the `NumericalValueVariationPoint` shall yield an integer, not a float.⌋

[7] The `min` and `max` values are defined optional, however in the 'Vendor Specific Module Definition' these values are mandatory.

Class	**EcucIntegerParamDef**			
Note	Configuration parameter type for Integer. This Class is only used by the AUTOSAR Classic Platform.			
Base	*ARObject*, *AtpDefinition*, *EcucCommonAttributes*, *EcucDefinitionElement*, *EcucParameterDef*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	EcucParamConfContainerDef.parameter			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
defaultValue	UnlimitedInteger	0..1	attr	Default value of the integer configuration parameter. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=codeGenerationTime
max	UnlimitedInteger	0..1	attr	Max value allowed for the parameter defined. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=codeGenerationTime
min	UnlimitedInteger	0..1	attr	Min value allowed for the parameter defined. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=codeGenerationTime

Table 2.11: EcucIntegerParamDef

Listing 2.10 shows the `EcuC Parameter Definition` XML file. The corresponding `ECU Configuration Values ARXML` file extract is shown in Listing 2.36.

```
<ECUC-INTEGER-PARAM-DEF>
  <SHORT-NAME>PositionInTask</SHORT-NAME>
  <DEFAULT-VALUE>0</DEFAULT-VALUE>
  <MAX>255</MAX>
  <MIN>0</MIN>
</ECUC-INTEGER-PARAM-DEF>
```

Listing 2.10: `EcucIntegerParamDef` Example

2.3.5.3 Float Type

[TPS_ECUC_02028] `EcucFloatParamDef` properties ⌈ To be able to specify parameters with floating number values the `EcucFloatParamDef` can be used. The additional attributes `min`, `max` and `defaultValue` can be specified as well[8].⌋

[TPS_ECUC_02115] Variable default value in `EcucFloatParamDef` ⌈ The attribute `defaultValue` of `EcucFloatParamDef` is subject to variant handling. The value can be computed using the variant handling mechanism.⌋

[TPS_ECUC_02117] Variable min, max values in `EcucFloatParamDef` ⌈ The attributes `min` and `max` of `EcucFloatParamDef` are subject to variant handling. The values can be computed using the variant handling mechanism.⌋

[TPS_ECUC_06033] Min and max values in `EcucFloatParamDef` ⌈ The `max` value shall be equal or bigger than the `min` value and the `min` value shall be equal or less than the `max` value.⌋

[8] The `min` and `max` values are defined optional, however in the 'Vendor Specific Module Definition' these values are mandatory.

[TPS_ECUC_06034] Special float values ⌈ The notation of the special float values "Not a Number" and positive/negative "infinity" shall be:

- NaN
- INF
- -INF

⌋

[TPS_ECUC_06087] INF and -INF allowed as defaultValue in EcucFloatParamDef ⌈The special float values INF and -INF are allowed to be specified as `defaultValue` of `EcucFloatParamDef`⌋

[TPS_ECUC_02075] Representation of `EcucFloatParamDef`s ⌈ For the representation the `IEEE double-precision 64-bit floating point` of the *IEEE 754-1985* standard [9] is used.⌋

Float values that exist on a target ECU which does not support 64 bit have to be converted to the nearest approximation of the value in float 32 for the target. In AUTOSAR XML the value shall be kept in 64 bit representation.

Class	**EcucFloatParamDef**			
Note	Configuration parameter type for Float. This Class is only used by the AUTOSAR Classic Platform.			
Base	*ARObject*, *AtpDefinition*, *EcucCommonAttributes*, *EcucDefinitionElement*, *EcucParameterDef*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	EcucParamConfContainerDef.parameter			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
defaultValue	Float	0..1	attr	Default value of the float configuration parameter. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=codeGenerationTime
max	Limit	0..1	attr	Max value allowed for the parameter defined. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=codeGenerationTime
min	Limit	0..1	attr	Min value allowed for the parameter defined. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=codeGenerationTime

Table 2.12: EcucFloatParamDef

[TPS_ECUC_06082] Definition of interval type for `EcucFloatParamDef.min` and `EcucFloatParamDef.max` ⌈The attributes `EcucFloatParamDef.min` and `EcucFloatParamDef.max` are used to define the usable interval of the respective `EcucFloatParamDef`. The interval itself may on both ends be defined as either

- closed: the provided value is included in the interval. This is expressed by setting the attribute `min.intervalType` resp. `max.intervalType` to `IntervalTypeEnum.closed`.

  - min.intervalType = closed is represented in an SWS (chapter: “Configuration Specifications”) by **prepending** the interval with “[”
  - max.intervalType = closed is represented in an SWS (chapter: “Configuration Specifications”) by **appending** the interval with “]”
  - Example: Range: [0,1] = (min.intervalType = closed / max.intervalType = closed) means the range shall be (>=0 and <= 1).
- open: the provided value in not included in the interval. This is expressed by setting the attribute min.intervalType resp. max.intervalType to IntervalTypeEnum.open.
  - min.intervalType = open is represented in an SWS (chapter: “Configuration Specifications”) by **prepending** the interval with “]”
  - max.intervalType = open is represented in an SWS (chapter: “Configuration Specifications”) by **appending** the interval with “[”
  - Example: Range:]0,1[= (min.intervalType = open / max.intervalType = open) means the range shall be (>0 and <1).

⌋

[TPS_ECUC_06083] Attribute EcucFloatParamDef.min.intervalType is not defined ⌈If the attribute min.intervalType is not defined then a closed interval is implicitly assumed for EcucFloatParamDef.min.⌋

[TPS_ECUC_06084] Attribute EcucFloatParamDef.max.intervalType is not defined ⌈If the attribute max.intervalType is not defined then a closed interval is implicitly assumed for EcucFloatParamDef.max.⌋

Listing 2.11 shows the ECUC Parameter definition XML file. The corresponding ECU Configuration Values ARXML file extract is shown in Listing 2.37.

```
<ECUC-FLOAT-PARAM-DEF>
  <SHORT-NAME>SchedulingPeriod</SHORT-NAME>
  <ORIGIN>AUTOSAR_ECUC</ORIGIN>
  <DEFAULT-VALUE>NaN</DEFAULT-VALUE>
  <MAX INTERVAL-TYPE="CLOSED">80</MAX>
  <MIN INTERVAL-TYPE="OPEN">0</MIN>
</ECUC-FLOAT-PARAM-DEF>
```

Listing 2.11: EcucFloatParamDef Example

2.3.5.4 String Parameter

[TPS_ECUC_02029] Subclasses of EcucAbstractStringParamDef ⌈ The subclasses of the class EcucAbstractStringParamDef provide means to specify

strings in the ECU Configuration Values. Additionally an optional defaultValue can be provided.⌋

[TPS_ECUC_02112] Variable default value in EcucAbstractStringParamDef ⌈ The attribute defaultValue of EcucAbstractStringParamDef and its subclasses is subject to variant handling. The value can be computed using the variant handling mechanism.⌋

[TPS_ECUC_06035] Regular expression ⌈ The regular expression is provided according to the Generic Structure Template [4].⌋

Class	«atpVariation» ***EcucAbstractStringParamDef*** (abstract)			
Note	Abstract class that is used to collect the common properties for StringParamDefs, LinkerSymbolDef, FunctionNameDef and MultilineStringParamDefs. **Tags:** vh.latestBindingTime=codeGenerationTime This Class is only used by the AUTOSAR Classic Platform.			
Base	*ARObject*, *AtpDefinition*, *EcucCommonAttributes*, *EcucDefinitionElement*, *EcucParameterDef*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	EcucFunctionNameDef, EcucLinkerSymbolDef, EcucMultilineStringParamDef, EcucStringParamDef			
Aggregated by	EcucParamConfContainerDef.parameter			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
defaultValue	VerbatimString	0..1	attr	Default value of the string configuration parameter.
maxLength	PositiveInteger	0..1	attr	Max length allowed for this string.
minLength	PositiveInteger	0..1	attr	Min length allowed for this string.
regular Expression	RegularExpression	0..1	attr	This represents the regular expression which shall be used to validate the string parameter value.

Table 2.13: EcucAbstractStringParamDef

Class	«atpVariation» **EcucStringParamDef**			
Note	Configuration parameter type for String. This Class is only used by the AUTOSAR Classic Platform.			
Base	*ARObject*, *AtpDefinition*, *EcucAbstractStringParamDef*, *EcucCommonAttributes*, *EcucDefinitionElement*, *EcucParameterDef*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	EcucParamConfContainerDef.parameter			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 2.14: EcucStringParamDef

Class	«atpVariation» **EcucMultilineStringParamDef**			
Note	Configuration parameter type for multiline Strings (including "carriage return"). This Class is only used by the AUTOSAR Classic Platform.			
Base	*ARObject*, *AtpDefinition*, *EcucAbstractStringParamDef*, *EcucCommonAttributes*, *EcucDefinitionElement*, *EcucParameterDef*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	EcucParamConfContainerDef.parameter			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 2.15: EcucMultilineStringParamDef

2.3.5.5 Linker Symbol Parameter

[TPS_ECUC_06006] `EcucLinkerSymbolDef` properties ⌈ When a parameter represents a linker symbol in the configured software the `EcucLinkerSymbolDef` shall be used. The actual values of the symbol defined will be specified by the implementing software and are not subject to configuration.⌋

[TPS_ECUC_02030] Programming language identifier limitations ⌈ The restriction on the `defaultValue` and the value of a `EcucLinkerSymbolDef` and its subclass are the common programming language identifier limitations: start with a letter or a special character (sc) followed by upper- and lower-case letters, digits and special characters:

```
identifier := (letter | sc) ( letter | digit | sc )*
```

where letter is [a-z] or [A-Z], sc is (_ | . | $ | %) and digit is [0-9].⌋

[TPS_ECUC_02031] Restriction on the length of `EcucLinkerSymbolDef` values and defaultValue ⌈ The restriction on the length of the default value and the value of a `EcucLinkerSymbolDef` is set to 255 characters.⌋

The class `EcucLinkerSymbolDef` does not introduce any additional attributes.

The `EcucLinkerSymbolDef` in fact represents the C-compiler symbol which later is translated into a linker symbol. With this element the usage of the `external` declaration of symbols (e.g. variables, constants) is possible.

Class	«atpVariation» **EcucLinkerSymbolDef**			
Note	Configuration parameter type for Linker Symbol Names like those used to specify memory locations of variables and constants. This Class is only used by the AUTOSAR Classic Platform.			
Base	*ARObject*, *AtpDefinition*, *EcucAbstractStringParamDef*, *EcucCommonAttributes*, *EcucDefinitionElement*, *EcucParameterDef*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	EcucParamConfContainerDef.parameter			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 2.16: EcucLinkerSymbolDef

Listing 2.12 shows the ECUC Parameter definition XML file. The corresponding `ECU Configuration Values ARXML` file extract is shown in Listing 2.32.

```
<ECUC-LINKER-SYMBOL-DEF>
  <SHORT-NAME>RtePimInitializationSymbol</SHORT-NAME>
  <ECUC-LINKER-SYMBOL-DEF-VARIANTS>
    <ECUC-LINKER-SYMBOL-DEF-CONDITIONAL>
      <DEFAULT-VALUE>MyPimInitValuesLightMaster</DEFAULT-VALUE>
    </ECUC-LINKER-SYMBOL-DEF-CONDITIONAL>
  </ECUC-LINKER-SYMBOL-DEF-VARIANTS>
</ECUC-LINKER-SYMBOL-DEF>
```

Listing 2.12: `EcucLinkerSymbolDef` Example

2.3.5.6 Function Name Parameter

[TPS_ECUC_02033] `EcucFunctionNameDef` properties ⌈ When a parameter represents a full function name in the configured software the `EcucFunctionNameDef` shall be used. With this feature functions (like callbacks) can be specified. The class `EcucFunctionNameDef` does not introduce any additional attributes.⌋

Class	«atpVariation» **EcucFunctionNameDef**			
Note	Configuration parameter type for Function Names like those used to specify callback functions. This Class is only used by the AUTOSAR Classic Platform.			
Base	*ARObject*, *AtpDefinition*, *EcucAbstractStringParamDef*, *EcucCommonAttributes*, *EcucDefinitionElement*, *EcucParameterDef*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	EcucParamConfContainerDef.parameter			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 2.17: EcucFunctionNameDef

Listing 2.13 shows the ECUC Parameter definition XML file. The corresponding `ECU Configuration Values ARXML` file extract is shown in Listing 2.33.

```
<ECUC-FUNCTION-NAME-DEF>
  <SHORT-NAME>EepJobEndNotification</SHORT-NAME>
  <ECUC-FUNCTION-NAME-DEF-VARIANTS>
    <ECUC-FUNCTION-NAME-DEF-CONDITIONAL>
      <DEFAULT-VALUE>Eep_JobEndNotification</DEFAULT-VALUE>
    </ECUC-FUNCTION-NAME-DEF-CONDITIONAL>
  </ECUC-FUNCTION-NAME-DEF-VARIANTS>
</ECUC-FUNCTION-NAME-DEF>
```

Listing 2.13: `EcucFunctionNameDef` Example

[TPS_ECUC_06075] `EcucFunctionNameDef` shall represent a valid C Identifier ⌈The defaultValue and the value of a `EcucFunctionNameDef` shall follow the pattern [a-zA-Z_][a-zA-Z0-9_]* defined in the context of the `CIdentifier`.⌋

2.3.5.7 Enumeration Parameter

[TPS_ECUC_02034] `EcucEnumerationParamDef` properties ⌈ When the parameter can be one choice of several possibilities the `EcucEnumerationParamDef` shall be used. It defines the parameter that will hold the actual value and may also define the `defaultValue` for the enumeration.⌋

The specification of variable default value for the enumeration is currently not supported.

Class	**EcucEnumerationParamDef**			
Note	Configuration parameter type for Enumeration. This Class is only used by the AUTOSAR Classic Platform.			
Base	*ARObject*, *AtpDefinition*, *EcucCommonAttributes*, *EcucDefinitionElement*, *EcucParameterDef*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	EcucParamConfContainerDef.parameter			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
defaultValue	Identifier	0..1	attr	Default value of the enumeration configuration parameter. This string needs to be one of the literals specified for this enumeration.
literal	EcucEnumerationLiteralDef	*	aggr	Aggregation on the literals used to define this enumeration parameter. This aggregation is optional if the surrounding EcucModuleDef has the category STANDARDIZED_MODULE_DEFINITION. If the category attribute of the EcucModuleDef is set to VENDOR_SPECIFIC_MODULE_DEFINITION then this aggregation is mandatory. **Tags:** atp.Splitkey=literal, literal.shortName

Table 2.18: EcucEnumerationParamDef

2.3.5.8 Enumeration Literal Definition

[TPS_ECUC_02035] Available choices of `EcucEnumerationParamDef`s are defined by aggregated `literal`s ⌈ To provide the available choices for the `EcucEnumerationParamDef` the `EcucEnumerationLiteralDef` is used. For each available choice there needs to be one `literal` defined.⌋

[TPS_ECUC_02036] The shortName of an `EcucEnumerationLiteralDef` is used to define the literal ⌈ For the text used to define the `EcucEnumerationLiteralDef` no additional attribute is needed because the `shortName` inherited from `Identifiable` is used to define the `literal`s.⌋

[TPS_ECUC_02054] Allowed literal strings ⌈ For the allowed string in `shortName` the restrictions apply as defined in the Generic Structure Template [4], in the primitive `Identifier`.⌋

This basically restricts the `shortName` to only containing the characters `[a-zA-Z][a-zA-Z0-9_]` and have a maximum length of 128 characters. If a more human readable text shall be provided the `longName` can be used which has much more freedom. This requires that configuration tools will show the optional `longName` to the users.

The relationship between the `EcucEnumerationParamDef` and the available `EcucEnumerationLiteralDef` is established using aggregations with the role name `literal` at the side of the `EcucEnumerationLiteralDef`.

[TPS_ECUC_02131] Origin information in literal definitions ⌈ Each `EcucEnumerationLiteralDef` has to provide information on its `origin`, which contains a string describing if the parameter is defined in the AUTOSAR standard (

'AUTOSAR_ECUC') or if the parameter is defined as a vendor specific parameter (e.g. 'VendorXYZ_v1.3').⌋

Class	**EcucEnumerationLiteralDef**			
Note	Configuration parameter type for enumeration literals definition. This Class is only used by the AUTOSAR Classic Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	EcucEnumerationParamDef.literal			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
ecucCond	EcucCondition Specification	0..1	aggr	If it evaluates to true the literal definition shall be processed as specified. Otherwise the literal definition shall be ignored. **Tags:** atp.Splitkey=ecucCond
origin	String	0..1	attr	String specifying if this literal is an AUTOSAR standardized literal or if the literal is vendor-specific.

Table 2.19: EcucEnumerationLiteralDef

Listing 2.14 shows the ECUC Parameter definition XML file. The corresponding ECU Configuration Values ARXML file extract is shown in Listing 2.34.

```
<ECUC-ENUMERATION-PARAM-DEF>
  <SHORT-NAME>RteGenerationMode</SHORT-NAME>
  <LITERALS>
    <ECUC-ENUMERATION-LITERAL-DEF>
      <SHORT-NAME>CompatibilityMode</SHORT-NAME>
      <LONG-NAME>
        <L-4 L="EN">Generate in Compatibility Mode</L-4>
      </LONG-NAME>
    </ECUC-ENUMERATION-LITERAL-DEF>
    <ECUC-ENUMERATION-LITERAL-DEF>
      <SHORT-NAME>VendorMode</SHORT-NAME>
      <LONG-NAME>
        <L-4 L="EN">Generate in Vendor Mode</L-4>
      </LONG-NAME>
    </ECUC-ENUMERATION-LITERAL-DEF>
  </LITERALS>
</ECUC-ENUMERATION-PARAM-DEF>
```

Listing 2.14: EcucEnumerationParamDef Example

2.3.5.9 AddInfo

[TPS_ECUC_02118] EcucAddInfoParamDef properties ⌈ The parameter EcucAddInfoParamDef is used to specify the need for formatted text in the ECU Configuration Values. The specification of the details on formatted text can be found in the AUTOSAR Generic Structure Template [4].⌋

Class	**EcucAddInfoParamDef**			
Note	Configuration Parameter Definition for the specification of formatted text in the ECU Configuration Parameter Description. This Class is only used by the AUTOSAR Classic Platform.			
Base	*ARObject*, *AtpDefinition*, *EcucCommonAttributes*, *EcucDefinitionElement*, *EcucParameterDef*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	EcucParamConfContainerDef.parameter			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 2.20: EcucAddInfoParamDef

Listing 2.15 shows the ECUC Parameter definition XML file. The corresponding ECU Configuration Values ARXML file extract is shown in Listing 2.38.

```
<ECUC-ADD-INFO-PARAM-DEF>
  <SHORT-NAME>DiagnosticTesterMessage</SHORT-NAME>
</ECUC-ADD-INFO-PARAM-DEF>
```

Listing 2.15: EcucAddInfoParamDef Example

2.3.6 References in Parameter Definition

There are five kinds of references available for the definition of configuration parameters referring to other entities.

- Reference to other configuration containers within the ECU Configuration Values (see section 2.3.6.1).
- A choice in the referenced configuration container can be specified and the ECU Configuration Values has the freedom (with restrictions) to choose to which target type the reference is pointing to (see section 2.3.6.2).
- Entities outside the ECU Configuration Values can be referenced when they have been specified in a different AUTOSAR Template (see section 2.3.6.3).
- Entities outside the ECU Configuration Values can be referenced using the instanceRef semantics defined in the Generic Structure Template [4] (see section 2.3.6.4).
- Reference to a destination that is specified via destinationUri (see section 2.3.6.6).

The meta-model of those references is shown in figure 2.12.

In the context of EcuC references, target type means the EcucContainerDef, which is the target of the the destination attribute reference (EcucReferenceDef.destination or EcucChoiceReferenceDef.destination) in the ECU Configuration Parameter Definition. In the ECU Configuration Values, an EcucReferenceValue can only refer to those EcucContainerValues, whose EcucContainerValue.definition (its type) matches the reference definition's destination attribute.

In other words: the reference definition's `destination` and the target value's `definition` have to refer to the same element.

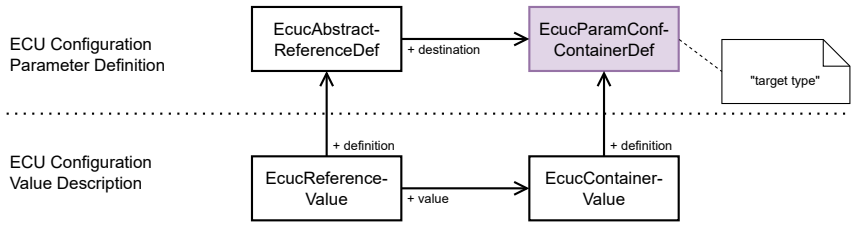

Figure 2.11: Definition of target type for references

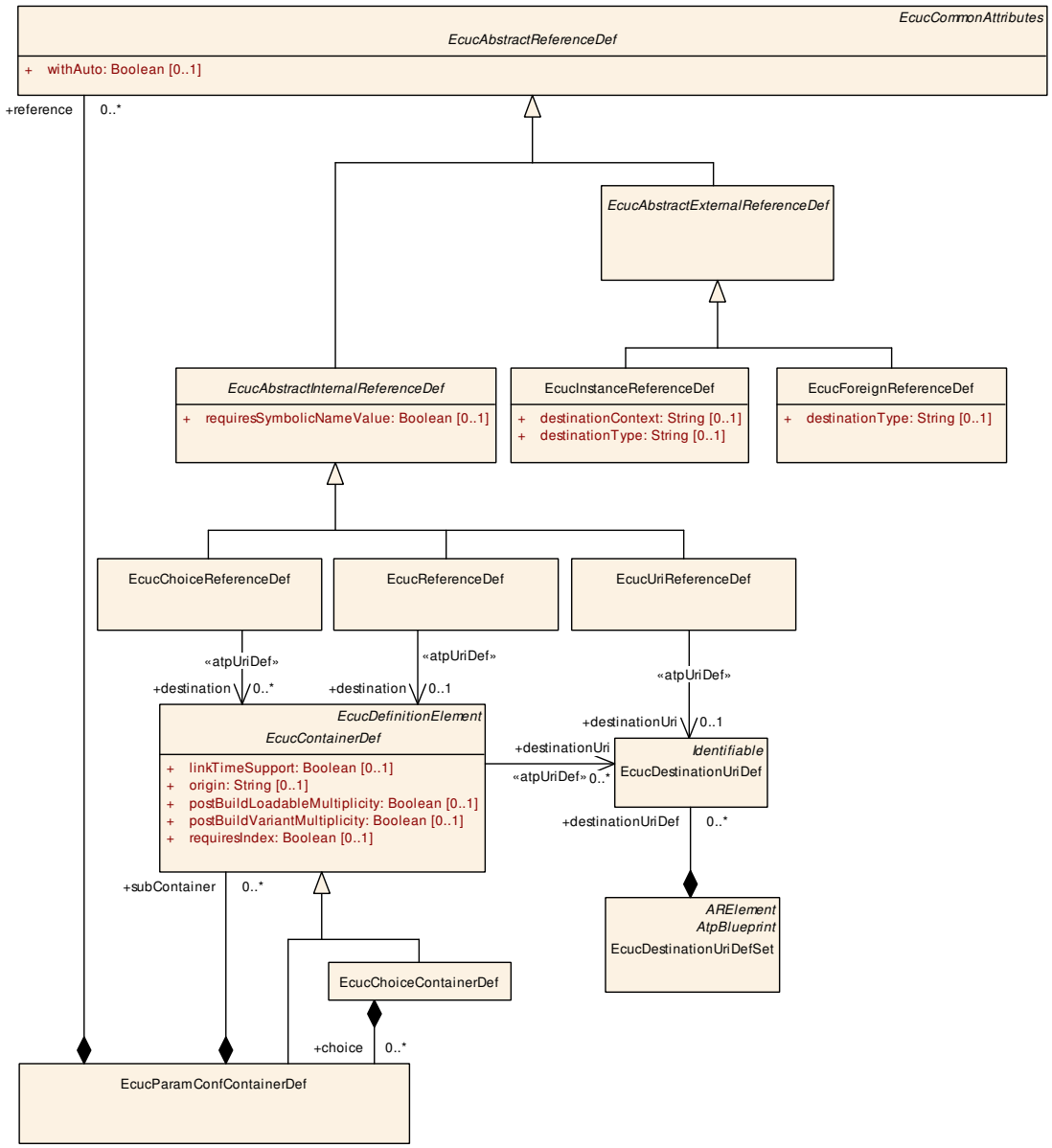

Figure 2.12: Class diagram for parameter references

[constr_3576] `EcucInstanceReferenceDef.destinationContext` always required

Imposition time: CP: `IT_EcucDef`

⌈The attribute `EcucInstanceReferenceDef.destinationContext` shall always be defined.⌋

[constr_3577] `EcucInstanceReferenceDef.destinationType` always required

Imposition time: CP: `IT_EcucDef`

⌈The attribute `EcucInstanceReferenceDef.destinationType` shall always be defined.⌋

[constr_3578] `EcucForeignReferenceDef.destinationType` always required

Imposition time: CP: `IT_EcucDef`

⌈The attribute `EcucForeignReferenceDef.destinationType` shall always be defined.⌋

[constr_3579] `EcucReferenceDef.destination` always required

Imposition time: CP: `IT_EcucDef`

⌈The attribute `EcucReferenceDef.destination` shall always be defined.⌋

[constr_3580] `EcucUriReferenceDef.destinationUri` always required

Imposition time: CP: `IT_EcucDef`

⌈The attribute `EcucUriReferenceDef.destinationUri` shall always be defined.⌋

[constr_3581] `EcucDestinationUriDefSet.destinationUriDef` always required

Imposition time: CP: `IT_EcucDef`

⌈The attribute `EcucDestinationUriDefSet.destinationUriDef` shall always be defined.⌋

[TPS_ECUC_02037] `EcucAbstractReferenceDef` properties ⌈ The abstract class `EcucAbstractReferenceDef` is used to specify the common parts of all reference definitions. `EcucAbstractReferenceDef` is an `Identifiable` so it is mandatory to give each reference definition a name. Also `EcucAbstractReferenceDef` is inheriting from `EcucDefinitionElement` so for each reference definition it can be specified how many such references might be present in the same configuration container later in the `ECU Configuration Values`.⌋

Class	***EcucAbstractReferenceDef*** (abstract)			
Note	Common class to gather the attributes for the definition of references. This Class is only used by the AUTOSAR Classic Platform.			
Base	*ARObject*, *AtpDefinition*, *EcucCommonAttributes*, *EcucDefinitionElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	*EcucAbstractExternalReferenceDef*, *EcucAbstractInternalReferenceDef*			
Aggregated by	EcucParamConfContainerDef.reference			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
withAuto	Boolean	0..1	attr	Specifies whether it shall be allowed on the value side to specify this reference value as "AUTO". If withAuto is "true" it shall be possible to set the "isAuto Value" attribute of the respective reference to "true". This means that the actual value will not be considered during ECU Configuration but will be (re-)calculated by the code generator and stored in the value attribute afterwards. These implicit updated values might require a re-generation of other modules which reference these values. If withAuto is "false" it shall not be possible to set the "is AutoValue" attribute of the respective reference to "true". If withAuto is not present the default is "false".

Table 2.21: EcucAbstractReferenceDef

Class	***EcucAbstractInternalReferenceDef*** (abstract)			
Note	Common abstract class to gather attributes for internal references (where the destination is located in the Ecu Configuration Description). This Class is only used by the AUTOSAR Classic Platform.			
Base	*ARObject*, *AtpDefinition*, *EcucAbstractReferenceDef*, *EcucCommonAttributes*, *EcucDefinitionElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	EcucChoiceReferenceDef, EcucReferenceDef, EcucUriReferenceDef			
Aggregated by	EcucParamConfContainerDef.reference			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
requires SymbolicName Value	Boolean	0..1	attr	If this attribute is set to true the implementation of the reference is done using a Symbolic Name defined by the referenced container according to TPS_ECUC_02108.

Table 2.22: EcucAbstractInternalReferenceDef

Class	***EcucAbstractExternalReferenceDef*** (abstract)			
Note	Common abstract class to gather attributes for external references (where the destination is not located in the ECU Configuration Description but in an another AUTOSAR Template). This Class is only used by the AUTOSAR Classic Platform.			
Base	*ARObject*, *AtpDefinition*, *EcucAbstractReferenceDef*, *EcucCommonAttributes*, *EcucDefinitionElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	EcucForeignReferenceDef, EcucInstanceReferenceDef			
Aggregated by	EcucParamConfContainerDef.reference			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 2.23: EcucAbstractExternalReferenceDef

2.3.6.1 Reference

[TPS_ECUC_02039] References between containers are established with the `EcucReferenceDef` ⌈ The `EcucReferenceDef` is used to establish references from one `EcucParamConfContainerDef` to one other specific `EcucParamConfContainerDef` or `EcucChoiceContainerDef` within the same `ECU Configuration Values`. For this purpose an object representing the reference has to be used.⌋

[TPS_ECUC_02038] Destination of `EcucReferenceDef` and `EcucChoiceReferenceDef`s is the `EcucContainerDef` ⌈ The `destination` for the `EcucReferenceDef` and the `EcucChoiceReferenceDef` is both the `EcucContainerDef`. So it is not possible to reference to a specific `EcucParameterDef`, `EcucReferenceDef` or `EcucModuleDef`.⌋

The reason is that there is no use-case where a direct reference to a parameter would be needed.

Class	**EcucReferenceDef**			
Note	Specify references within the ECU Configuration Description between parameter containers. This Class is only used by the AUTOSAR Classic Platform.			
Base	*ARObject*, *AtpDefinition*, *EcucAbstractInternalReferenceDef*, *EcucAbstractReferenceDef*, *EcucCommonAttributes*, *EcucDefinitionElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	EcucParamConfContainerDef.reference			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
destination	EcucContainerDef	0..1	ref	Exactly one reference to a parameter container is allowed as destination. **Stereotypes:** atpUriDef **Tags:** atp.Splitkey=destination

Table 2.24: EcucReferenceDef

The role name at the `EcucReferenceDef` has to be `reference` and the role name at the referenced container has to be `destination` (see figure 2.13 for an example).

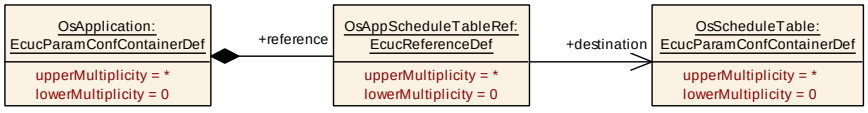

Figure 2.13: Example of an object diagram for a reference

In the example in figure 2.13 the 'OsApplication' is defined to contain references to the 'OsScheduleTable'. The references are called 'OsAppScheduleTableRef' and there can be several such references in the actual `ECU Configuration Values` document. For the multiplicity of references the multiplicity definition on the `EcucReferenceDef` are relevant (in the example the `lowerMultiplicity` is '0' and the `upperMultiplicity` is '*'). The multiplicity of the referenced container is not considered for references.

In the ECU Configuration Parameter Definition XML file the `destination` has to be identified unambiguously because the names of configuration parameters are not required to be unique throughout the whole ECU Configuration Parameter Definition. So

there might be a parameter defined in the CAN-Driver with the same name as one parameter defined in the ADC-Driver. For this reason the containment hierarchy of the referenced configuration parameter has to be denoted in the definition XML file, as shown in Listing 2.16. In this example the referenced parameter will be found in the definition of the Os module directly in the AUTOSARParameterDefinition. The corresponding ECU Configuration Values ARXML file extract is shown in Listing 2.39.

```
<ECUC-REFERENCE-DEF>
        <SHORT-NAME>OsAppScheduleTableRef</SHORT-NAME>
        <DESTINATION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF">/AUTOSAR/
            EcucDefs/Os/OsScheduleTable</DESTINATION-REF>
</ECUC-REFERENCE-DEF>
```

Listing 2.16: EcucReferenceDef Example

2.3.6.2 Choice Reference

[TPS_ECUC_02040] EcucChoiceReferenceDef properties ⌈ With the EcucChoiceReferenceDef it is possible to define one reference where the destination is specified to be one of several possible kinds. To be able to define such a choice an object of the class EcucChoiceReferenceDef has to be aggregated in a container with the role name reference at the EcucChoiceReferenceDef object. The destinations of a EcucChoiceReferenceDef may be EcucParamConfContainerDef and EcucChoiceContainerDef.⌋

Class	**EcucChoiceReferenceDef**			
Note	Specify alternative references where in the ECU Configuration description only one of the specified references will actually be used. This Class is only used by the AUTOSAR Classic Platform.			
Base	*ARObject*, *AtpDefinition*, *EcucAbstractInternalReferenceDef*, *EcucAbstractReferenceDef*, *EcucCommonAttributes*, *EcucDefinitionElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	EcucParamConfContainerDef.reference			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
destination	EcucContainerDef	*	ref	All the possible parameter containers for the reference are specified. **Stereotypes:** atpUriDef **Tags:** atp.Splitkey=destination

Table 2.25: EcucChoiceReferenceDef

All the available choices are connected via associations with the role name destination at the referenced object (see example in figure 2.14).

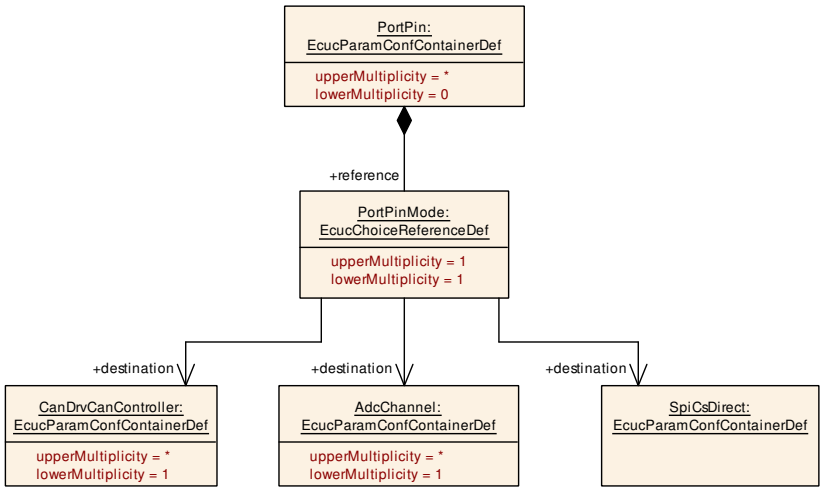

Figure 2.14: Example of an object diagram for a choice reference

In this example an actual instance of the 'PortPinMode' container can reference one of the three defined containers. Once again the multiplicity is defined by the `EcucChoiceReferenceDef` (here the default '1' for lower and upper) and the multiplicities of the referenced containers are not relevant for choice references.

Also the destination needs to be defined unambiguously in the ECU Configuration Parameter Definition XML file like shown in Listing 2.17. The corresponding `ECU Configuration Values ARXML` file extract is shown in Listing 2.40.

```
<ECUC-CHOICE-REFERENCE-DEF>
  <SHORT-NAME>PortPinMode</SHORT-NAME>
  <DESTINATION-REFS>
    <DESTINATION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF">/AUTOSAR/EcucDefs
        /Can/CanConfigSet</DESTINATION-REF>
    <DESTINATION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF">/AUTOSAR/EcucDefs
        /Adc/AdcConfigSet</DESTINATION-REF>
    <DESTINATION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF">/AUTOSAR/EcucDefs
        /Spi/SpiCsDirect</DESTINATION-REF>
  </DESTINATION-REFS>
</ECUC-CHOICE-REFERENCE-DEF>
```

Listing 2.17: `EcucChoiceReferenceDef` Example

In the `ECU Configuration Values`, the actual choice will be taken and there will be only one reference destination left[9].

[9]The `EcucDefinitionElement` is used to specify the possible occurrences of each reference later in the `ECU Configuration Values`. The `EcucChoiceReferenceDef` specifies multiple possible destinations for one reference but later in the `ECU Configuration Values`, there can only be exactly one destination described. So the freedom of multiple destinations is only available on the definition of references, if several containers need to be referenced the `EcucDefinitionElement` has to be set to more than 1, even for the `EcucChoiceReferenceDef`.

2.3.6.3 Foreign Reference

[TPS_ECUC_02041] `EcucForeignReferenceDef` properties ⌈ To be able to refer to descriptions of other AUTOSAR templates the parameter definition `EcucForeignReferenceDef` is used. With the attribute `destinationType` the type of the referenced entity has to be specified.⌋

Class	**EcucForeignReferenceDef**			
Note	Specify a reference to an XML description of an entity described in another AUTOSAR template. This Class is only used by the AUTOSAR Classic Platform.			
Base	*ARObject*, *AtpDefinition*, *EcucAbstractExternalReferenceDef*, *EcucAbstractReferenceDef*, *EcucCommonAttributes*, *EcucDefinitionElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	EcucParamConfContainerDef.reference			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
destinationType	String	0..1	attr	The type in the AUTOSAR Metamodel to which instance this reference is allowed to point to.

Table 2.26: EcucForeignReferenceDef

[TPS_ECUC_02042] Specification of the `destinationType` in a `EcucForeignReferenceDef` ⌈ Since the AUTOSAR Schema generator rules require the class names of all `Referrable`s to be unique within the AUTOSAR 'M2:: AUTOSAR Templates' meta-model, it is sufficient to provide only the actual class name of the referenced class in the `destinationType`.⌋

Note: This is shown in Listing 2.18

[TPS_ECUC_06088] Specification of the `destinationType` format in a `EcucForeignReferenceDef` ⌈The string entered as `destinationType` shall have the name of a M2 class defined in the meta-model [8] under 'M2:: AUTOSAR Templates' as it is represented in the XML-Schema [10] and the referenced class needs to be derived (directly or indirectly) from `Referrable`. In the generated Parameter Definition XML file [7] the XML-Schema name shall be used.⌋

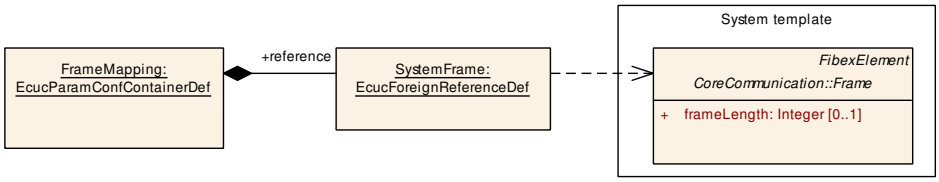

Figure 2.15: Example of an object diagram for a foreign reference

In the example in figure 2.15 the reference is defined to be pointing to a description of a `Frame`. The `Frame` is defined in the System Template meta-model [1] and is derived from `Identifiable` (which is a subclass of `Referrable`). The corresponding `ECU Configuration Values ARXML` file extract is shown in Listing 2.41.

```
<ECUC-FOREIGN-REFERENCE-DEF>
  <SHORT-NAME>SystemFrame</SHORT-NAME>
  <DESTINATION-TYPE>FRAME</DESTINATION-TYPE>
```

```
</ECUC-FOREIGN-REFERENCE-DEF>
```

Listing 2.18: `EcucForeignReferenceDef` Example

A foreign reference is intended to be used when no specific reference exists. While it would, for example, be possible to create an `EcucForeignReferenceDef` to an `EcucContainerValue` (or even an `EcucParamConfContainerDef`), the specific type `EcucReferenceDef` is better suited to this task, and should be used instead.

In general, the classes of the ECU configuration metamodel are out of scope for `EcucForeignReferenceDef.destinationType`, with one notable exception: There is a relatively rare usecase to refer to the EcuConfiguration of a complete BSW module (for example as a stand-in for that BSW module). For this, no specific reference type was created, so instead an `EcucForeignReferenceDef` is used, with the `destinationType` set to ECUC-MODULE-CONFIGURATION-VALUES. One example is the `EcucPduDedicatedPartition`, which allows mapping a `Pdu` to a `EcucPartition`, when it is used by a specific BSW module (i.e. in the context of that module). To be able to specify a BSW module, the foreign reference `EcucPduDedicatedPartitionBswModuleRef` targets the `EcucModuleConfigurationValues`.

2.3.6.4 Instance Reference

[TPS_ECUC_02060] `EcucInstanceReferenceDef` properties ⌈ To be able to reference to descriptions of other AUTOSAR templates with the `instanceRef` semantics[10] the parameter definition `EcucInstanceReferenceDef` is used. With the attribute `destinationType` the type of the referenced entity has to be specified. With the attribute `destinationContext` the context expression has to be specified.⌋

[TPS_ECUC_02082] Specification of the `destinationType` in a `EcucInstanceReferenceDef` ⌈ The string entered as `destinationType` shall have the name of a M2 class defined in the meta-model [8] under 'M2::AUTOSAR Templates' as it is represented in the XML-Schema [10] and the referenced class needs to be derived (directly or indirectly) from `Referrable`. In the generated Parameter Definition XML file [7] the XML-Schema name shall be used.⌋

[TPS_ECUC_02083] Specification of the `destinationContext` in a `EcucInstanceReferenceDef` ⌈ The string entered as `destinationContext` shall be an ordered list of M2 class names defined in the meta-model [8] under 'M2::AUTOSAR Templates' as it is represented in the XML schema [10] separated by the SPACE character. Additionally the '*' character can be used to indicate none or multiple occurrence of the M2 class BEFORE the '*' character.⌋

Examples of `destinationContext` expressions are:

[10] For a detailed description of the `instanceRef` concept please refer to the Generic Structure Template [4]

```
  SW-COMPONENT-PROTOTYPE R-PORT-PROTOTYPE

  ROOT-SW-COMPOSITION-PROTOTYPE SW-COMPONENT-PROTOTYPE PORT-PROTOTYPE

  ROOT-SW-COMPOSITION-PROTOTYPE SW-COMPONENT-PROTOTYPE PORT-PROTOTYPE
DATA-PROTOTYPE*
```

Class	**EcucInstanceReferenceDef**			
Note	Specify a reference to an XML description of an entity described in another AUTOSAR template using the INSTANCE REFERENCE semantics. This Class is only used by the AUTOSAR Classic Platform.			
Base	*ARObject*, *AtpDefinition*, *EcucAbstractExternalReferenceDef*, *EcucAbstractReferenceDef*, *EcucCommonAttributes*, *EcucDefinitionElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	EcucParamConfContainerDef.reference			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
destination Context	String	0..1	attr	The context in the AUTOSAR Metamodel to which' this reference is allowed to point to.
destinationType	String	0..1	attr	The type in the AUTOSAR Metamodel to which' instance this reference is allowed to point to.

Table 2.27: EcucInstanceReferenceDef

[TPS_ECUC_02061] Specification of the destinationType in a EcucInstanceReferenceDef ⌈ Since the AUTOSAR Schema generator rules require the class names of all `Referrable`s to be unique within the AUTOSAR 'M2:: AUTOSAR Templates' metamodel, it is sufficient to provide only the actual class name of the referenced class in the `destinationType`.⌋

Note: This is shown in Listing 2.19

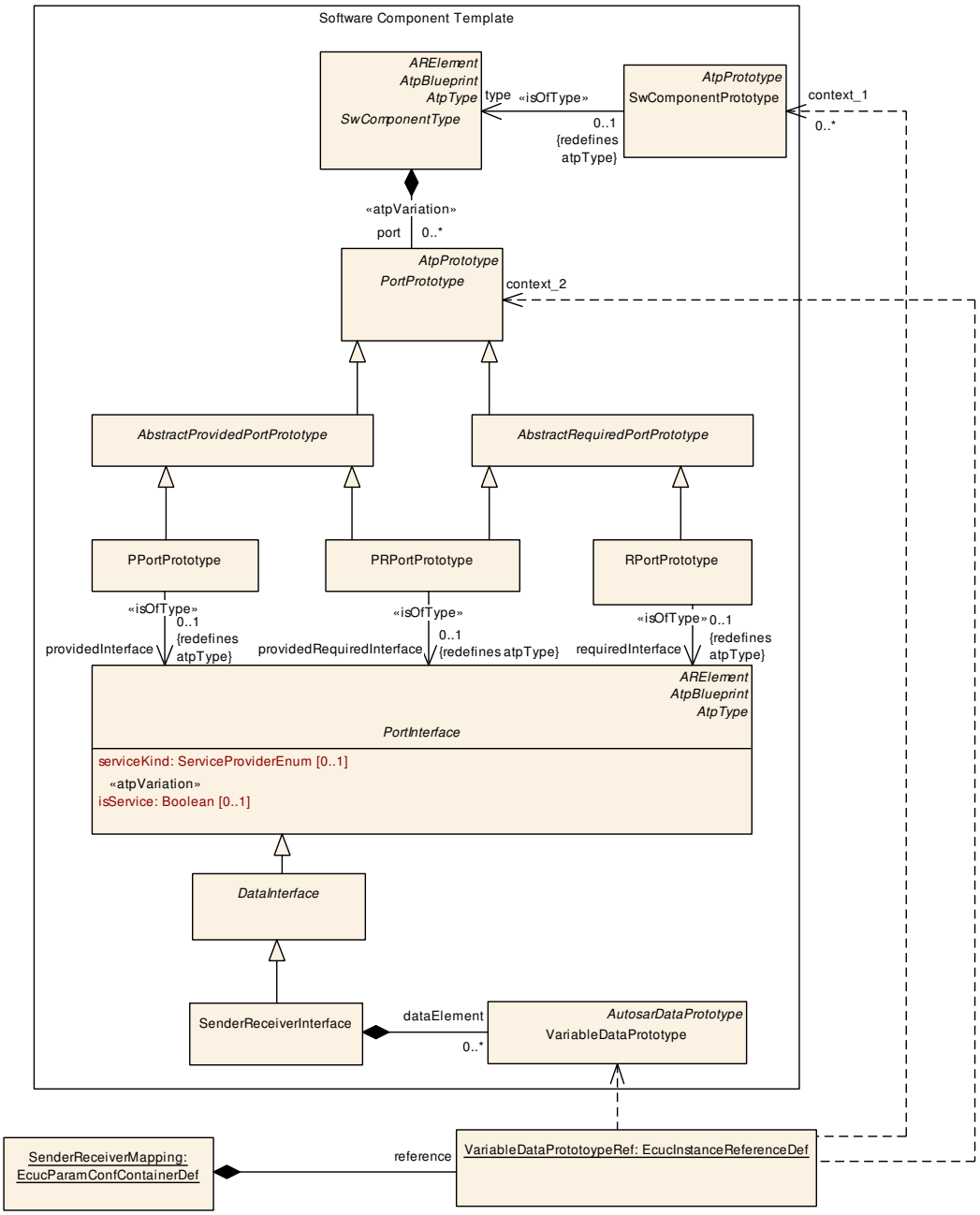

Figure 2.16: Example of an object diagram for an instance reference

In the example in figure 2.16 the reference is defined to be pointing to a description of a 'VARIABLE-DATA-PROTOTYPE'. The 'VARIABLE-DATA-PROTOTYPE' is defined in the Software Component Template meta-model [11] and is derived from `Identifiable` (which is a subclass of `Referrable`). Via the `destinationContext` it is specified that each 'VARIABLE-DATA-PROTOTYPE' exists in the context of a 'PORT-PROTOTYPE', which itself is in the context of the 'SW-COMPONENT-PROTOTYPE'. The corresponding `ECU Configuration Values ARXML` file extract is shown in Listing 2.42.

```
<ECUC-INSTANCE-REFERENCE-DEF>
  <SHORT-NAME>VariableDataPrototypeRef</SHORT-NAME>
```

```
  <DESTINATION-CONTEXT>SW-COMPONENT-PROTOTYPE* PORT-PROTOTYPE</DESTINATION-
     CONTEXT>
  <DESTINATION-TYPE>VARIABLE-DATA-PROTOTYPE</DESTINATION-TYPE>
</ECUC-INSTANCE-REFERENCE-DEF>
```

Listing 2.19: `EcucInstanceReferenceDef` Example

Although the ECU Configuration Parameter Definition of the EcucForeign-ReferenceDef and EcucInstanceReferenceDef are similar there is a difference how those references are represented in the ECU Configuration Values (see section 2.4.5).

2.3.6.5 Symbolic Name Reference

[TPS_ECUC_02146] Symbolic Name Reference properties ⌈An EcucAbstract-InternalReferenceDef with the attribute requiresSymbolicNameValue set to true is used to establish the relationship between the user of a symbolic name and the provider of a symbolic name. The object defining the EcucAbstractInternal-ReferenceDef is the user and the destination of the reference is the provider of the symbolic name.⌋

[TPS_ECUC_02063] Parameters with `symbolicNameValue` = true ⌈ If the attribute symbolicNameValue of a configuration parameter (see [TPS_ECUC_02014]) is set to true this configuration parameter contributes to the value of the symbolic name. Only one configuration parameter within a container may have the attribute symbolic-NameValue set to true.⌋

If the attribute symbolicNameValue is not present it shall be assumed to be set to false.

[constr_5520] Latest configuration time to change the values of `symbolicName-Value` parameters

Imposition time: CP: IT_EcucDef

⌈The latest configuration time to change the values of EcucParameterDefs with symbolicNameValue attribute set to true shall be PreCompile with the setting of postBuildLoadableValue and postBuildVariantValue to false.⌋

[constr_5521] Latest configuration time to change the number of instances of `symbolicNameValue` parameters ⌈The latest configuration time to change the number of instances of EcucParameterDefs with symbolicNameValue attribute set to true shall be PreCompile with the setting of postBuildLoadableMultiplicity and postBuildVariantMultiplicity to false.⌋

In the example definition shown in figure 2.17 the CorTst module can contain a CorTstDemEventParameterRefs. Those errors need to be defined in the Dem module. And only the Dem module is able to define actual numbers associated with these errors when all errors have been specified and collected in the Dem module. Those asso-

ciated values can be stored in the DemEventId parameter which belongs to each DemEventParameter.

For an example how this is used in the ECU Configuration Values refer to section 2.4.5.2. The corresponding ECU Configuration Values ARXML file extract is shown in Listing 2.43.

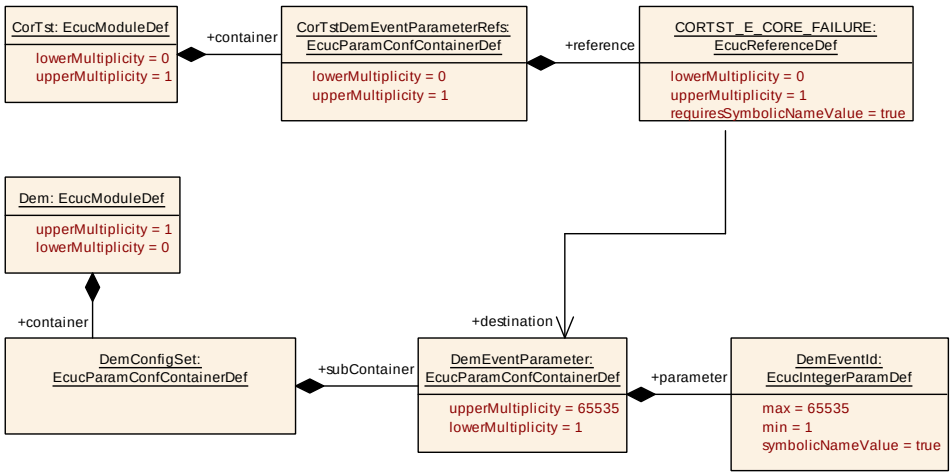

Figure 2.17: Example of an object diagram for a Symbolic Name Reference

```
<ECUC-MODULE-DEF>
  <SHORT-NAME>CorTst</SHORT-NAME>
  <CONTAINERS>
    <ECUC-PARAM-CONF-CONTAINER-DEF>
      <SHORT-NAME>CorTstDemEventParameterRefs</SHORT-NAME>
      <LOWER-MULTIPLICITY>0</LOWER-MULTIPLICITY>
      <UPPER-MULTIPLICITY>1</UPPER-MULTIPLICITY>
      <REFERENCES>
        <ECUC-REFERENCE-DEF>
          <SHORT-NAME>CORTST_E_CORE_FAILURE</SHORT-NAME>
          <LOWER-MULTIPLICITY>1</LOWER-MULTIPLICITY>
          <UPPER-MULTIPLICITY>1</UPPER-MULTIPLICITY>
          <REQUIRES-SYMBOLIC-NAME-VALUE>true</REQUIRES-SYMBOLIC-NAME-VALUE>
          <DESTINATION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF">/AUTOSAR/
              EcucDefs/Dem/DemEventParameter</DESTINATION-REF>
        </ECUC-REFERENCE-DEF>
      </REFERENCES>
    </ECUC-PARAM-CONF-CONTAINER-DEF>
  </CONTAINERS>
</ECUC-MODULE-DEF>
<ECUC-MODULE-DEF>
  <SHORT-NAME>Dem</SHORT-NAME>
  <CONTAINERS>
    <ECUC-PARAM-CONF-CONTAINER-DEF>
      <SHORT-NAME>DemEventParameter</SHORT-NAME>
      <LOWER-MULTIPLICITY>0</LOWER-MULTIPLICITY>
      <UPPER-MULTIPLICITY-INFINITE>true</UPPER-MULTIPLICITY-INFINITE>
      <PARAMETERS>
        <ECUC-INTEGER-PARAM-DEF>
          <SHORT-NAME>DemEventId</SHORT-NAME>
          <LOWER-MULTIPLICITY>1</LOWER-MULTIPLICITY>
```

```
          <UPPER-MULTIPLICITY>1</UPPER-MULTIPLICITY>
          <SYMBOLIC-NAME-VALUE>true</SYMBOLIC-NAME-VALUE>
        </ECUC-INTEGER-PARAM-DEF>
      </PARAMETERS>
    </ECUC-PARAM-CONF-CONTAINER-DEF>
  </CONTAINERS>
</ECUC-MODULE-DEF>
```

Listing 2.20: Symbolic Name Reference Example

2.3.6.6 Uri Reference

2.3.6.6.1 Overview

Chapters 2.3.6.1 Reference and 2.3.6.2 Choice Reference showed different kinds of references, where the `target type` was known at the time when the `ECU Configuration Parameter Definition` was created.

In other words, on the parameter definition side, the owner of the reference has a direct knowledge about the target of the reference (the `target type`). The `target type` could not be exchanged without the owner of the reference taking notice.

In Figure 2.18, the topmost row represents the conceptual idea. There is an element that owns a reference (examples outside of AUTOSAR could be a table in a database, or a class), the reference itself (a column in a database that contains a foreign id of another table, a member variable / field in a class whose type is a reference to another class), drawn as an arrow, and the `target type` (the other table, the other class). The second row shows how this concept is implemented in EcuC on the parameter definition side, the third row shows the EcuC value side.

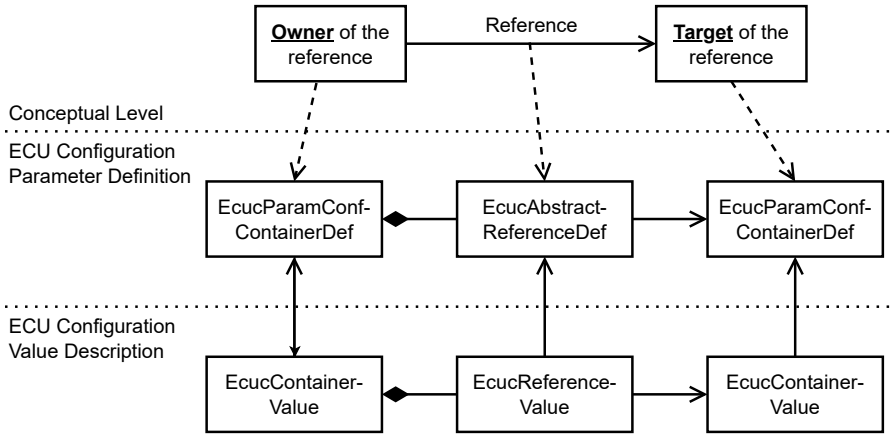

Figure 2.18: Conceptual idea of a reference and the implementation in EcuC

While this is sufficient in the vast majority of cases, there are some specific use cases, where it is necessary to introduce a more flexible mechanism for creating a reference.

Known use cases for an extended reference mechanism are:

- The position of the `target type` is unknown to the owner of the reference.
- The owner of the reference does not know in advance how many `target type`s for the reference exist.
- The `target type` of the reference needs to be replaced without the necessity to update the owner of the reference.

The solution for these requirements is to not base the reference on the `shortName` path of the reference target. Instead, the reference target and the reference owner get together via an URI-based indirection mechanism, such that the **match is created if both the owner of the reference and the reference target refer to the identical URI** (and fulfill further conditions described below).

This approach is sketched in Figure 2.19. Figure 2.20 shows an example, where the policy has one child container, which has a child parameter. Both the definiton and value side consist of two elements: one container with one child parameter.

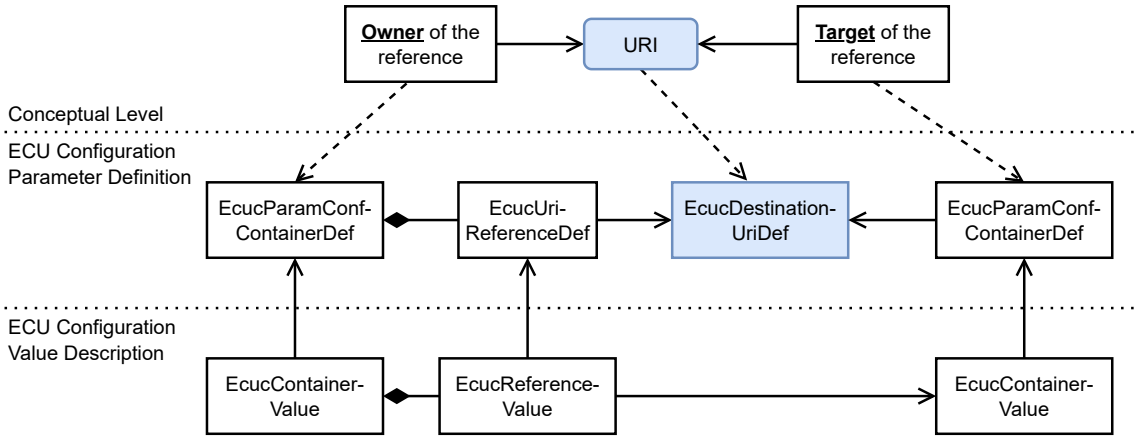

Figure 2.19: Conceptual idea of an URI-based reference and the implementation in EcuC

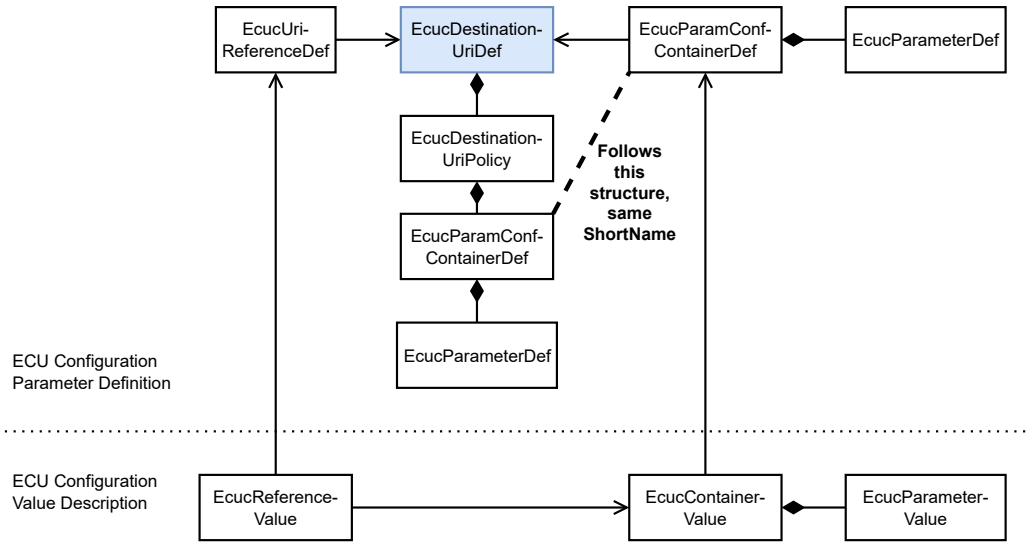

Figure 2.20: Structure for Uri Reference

Please note that several target types may reference the same URI. The owner of the URI-based reference does not even know how many target types exist.

Because the target type of the reference is not known to the owner of the reference, it is typically necessary to at least formulate some constraints to the structure and content of the target type.

These structural constraints can neither be defined in the context of the owner of the reference, nor in the context of the reference target. Instead, the constraints need to be formulated in the context of the definition of the URI that is the basis for the extended referencing mechanism. The following constraints may apply:

- the target container shall aggregate certain sub-containers with given shortNames (which in turn can aggregate containers, parameters and references)
- the target container shall aggregate certain parameters with given shortNames and given multiplicities.
- the target container shall aggregate certain references with a given shortNames and given multiplicities.

The definition of a URI is represented by meta-class EcucDestinationUriDef. This means that all model elements that want to participate in a URI-based reference definition need to reference the same EcucDestinationUriDef to indicate this relation.

The model elements involved in the usage of a URI-based reference are:

- An EcucContainerDef references an EcucDestinationUriDef in the role destinationUri, in order to indicate that it wants to become the target of an URI-based reference.
- The owner of the URI-based reference defines an EcucUriReferenceDef, which references EcucDestinationUriDef in the role destinationUri.

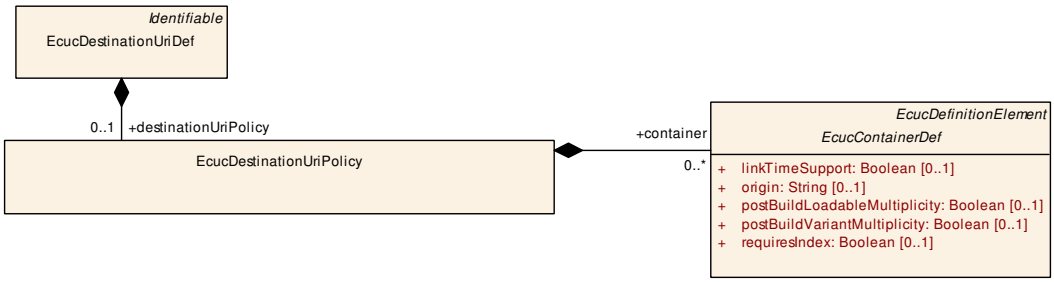

Figure 2.21: EcucDestinationUriDef details

The reference EcucContainerDef.destinationUri

Please note that there is a reference in the role EcucContainerDef.destinationUri to EcucDestinationUriDef. This reference is not shown in Figure 2.21 on purpose, because it does **NOT** mean that an EcucContainerDef aggregated in the role EcucDestinationUriPolicy.container needs to define a destinationUri.

Instead, as explained earlier, this reference is used by a regular EcucContainerDef (one that is not aggregated by an EcucDestinationUriPolicy) to indicate that it is a valid the target of an URI-based reference.

This reference would not make any sense, if it were used in the context of an EcucContainerDef, which is aggregated in the role EcucDestinationUriPolicy.container. After all, if this reference existed, it would only reference the owner of the EcucContainerDef and, therefore, **indicate a relation that is already defined by the aggregation chain**.

[constr_13000] Existence of reference in the role EcucContainerDef.destinationUri for an EcucContainerDef aggregated in the role EcucDestinationUriPolicy.container

Imposition time: CP: IT_EcucDef

⌈If an EcucContainerDef is **exclusively** aggregated in the role EcucDestinationUriPolicy.container, then it shall not define a reference in the role destinationUri.⌋

An Analogy

As just explained, the URI-based reference mechanism allows to define a reference to any target type, which follows a defined structure. As an analogy, consider object oriented design, where you have an interface If_A that defines a method doA(). A class Impl_B implementing If_A has to provide an implementation for doA(). Another class UserC can define a field myField of type If_A. If, at runtime, myField is set to an instance of Impl_B, then calling myField.doA() will dispatch to Impl_B.doA().

On the definition side, the URI-based reference mechanism consists of three parts:

- the "interface" - EcucDestinationUriDef - and its structure - defined via a EcucDestinationUriPolicy aggregated as destinationUriPolicy
- a "class implementing the interface" - a valid reference target that conforms to the interface, i.e. an EcucContainerDef that references an EcucDestinationUriDef via destinationUri
- a "reference to the interface" - an EcucUriReferenceDef whose destinationUri refers to the EcucDestinationUriDef

On the value side, the reference works the same way as any other reference to an EcucContainerValue:

- the target - an EcucContainerValue, whose definition points to the EcucContainerDef that references the EcucDestinationUriDef
- the reference EcucReferenceValue, whose definition points to the EcucUriReferenceDef

Previous versions of the standard provided some flexibility in the structure of the target type, using so called nesting contracts. Those were configured via EcucDestinationUriPolicy.destinationUriNestingContract. To simplify the usage of URI-based references, this was removed. All URI-based references now behave like the previous setting targetContainer.

In terms of structure, the target type itself has to fulfill the EcucDestinationUriPolicy. It has the same shortName and multiplity (lowerMultiplicity, upperMultiplicity, upperMultiplicityInfinite) as the EcucDestinationUriPolicy's only child.

2.3.6.6.2 Detailed Description of URI-based References

[TPS_ECUC_01050] Semantics of EcucUriReferenceDef ⌈Meta-class EcucUriReferenceDef is used to support the definition of an URI-based reference. Detailed constraints regarding the possible targets of the URI-based reference are specified in the scope of the aggregation of EcucDestinationUriPolicy in the role destinationUriPolicy.

The existence of an URI-base reference is established by

- an EcucContainerDef that references EcucDestinationUriDef in the role destinationUri in order to **indicate that it wants to become the target of an URI-based reference** and
- an EcucUriReferenceDef that references EcucDestinationUriDef in the role destinationUri to represent **the owner of the URI-based reference**.

⌋

Please note that an EcucContainerDef can define several destinationUris and therefore become a potential target for several EcucUriReferenceDefs. With the EcucUriReferenceDef it is possible to define a reference to containers in different modules independent from the concrete definition of the target container.

Class	**EcucUriReferenceDef**			
Note	Definition of reference with a destination that is specified via a destinationUri. With such a reference it is possible to define a reference to a EcucContainerDef in a different module independent from the concrete definition of the target container. This Class is only used by the AUTOSAR Classic Platform.			
Base	*ARObject*, *AtpDefinition*, *EcucAbstractInternalReferenceDef*, *EcucAbstractReferenceDef*, *EcucCommonAttributes*, *EcucDefinitionElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	EcucParamConfContainerDef.reference			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	EcucUriReferenceDef			
destinationUri	EcucDestinationUriDef	0..1	ref	Any EcucContainerDef with a destinationUri that is identical to the destinationUri that is referenced here defines a valid target. **Stereotypes:** atpUriDef **Tags:** atp.Splitkey=destinationUri

Table 2.28: EcucUriReferenceDef

Class	EcucDestinationUriDefSet			
Note	This class represents a list of EcucDestinationUriDefs. **Tags:** atp.recommendedPackage=EcucDestinationUriDefSets This Class is only used by the AUTOSAR Classic Platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
destinationUriDef	EcucDestinationUriDef	*	aggr	This is one particular EcucDestinationUriDef. **Tags:** atp.Splitkey=destinationUriDef, destinationUriDef.shortName

Table 2.29: EcucDestinationUriDefSet

Class	EcucDestinationUriDef			
Note	Description of an EcucDestinationUriDef that is used as target of EcucUriReferenceDefs. This Class is only used by the AUTOSAR Classic Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	EcucDestinationUriDefSet.destinationUriDef			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
destinationUriPolicy	EcucDestinationUriPolicy	0..1	aggr	Description of the targeted EcucContainerDef. **Tags:** atp.Splitkey=destinationUriPolicy

Table 2.30: EcucDestinationUriDef

[constr_3582] `EcucDestinationUriDef.destinationUriPolicy` always required

Imposition time: CP: `IT_EcucDef`

⌈The attribute `EcucDestinationUriDef.destinationUriPolicy` shall always be defined.⌋

Class	EcucDestinationUriPolicy
Note	The EcucDestinationUriPolicy describes the EcucContainerDef that will be targeted by EcucUriReference Defs. The type of the description is dependent of the destinationUriNestingContract attribute. This Class is only used by the AUTOSAR Classic Platform.
Base	*ARObject*
Aggregated by	EcucDestinationUriDef.destinationUriPolicy

▽

△

Class	**EcucDestinationUriPolicy**			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
container	EcucContainerDef	*	aggr	Description of the targetContainer in case that the destinationUriNestingPolicy is set to targetContainer. In all other cases the subContainers of the target container are defined here. **Tags:** atp.Splitkey=container, container.shortName

Table 2.31: EcucDestinationUriPolicy

[TPS_ECUC_01052] Target type implements `EcucDestinationUriPolicy.container` ⌈The attribute `EcucDestinationUriPolicy.container` is expressing the constraints applicable for an `EcucContainerDef` that wants to become a target for the enclosing `EcucDestinationUriDef`.

For this purpose, the `EcucContainerDef` needs to follow the structure of `EcucDestinationUriPolicy.container`:

- Specific primitive attributes of the `EcucContainerDef`, in particular
  - `shortName`
  - `lowerMultiplicity` and one of
  - `upperMultiplicity` or
  - `upperMultiplicityInfinite`

  need to be identical to `EcucDestinationUriPolicy.container`.
- Specific aggregations of the `EcucContainerDef`, in particular
  - `EcucParamConfContainerDef.parameter`
  - `EcucParamConfContainerDef.reference`
  - `EcucParamConfContainerDef.subContainer`

  for which corresponding elements (identified by a matching pair of `shortName`s) in the context of the `EcucDestinationUriPolicy.container` exist, shall be identical to the applicable aggregations of `EcucDestinationUriPolicy.container`. This rule may be applicable in a recursive manner if `EcucDestinationUriPolicy.container.subContainer` exists.

It is allowed that the `EcucContainerDef` (the target of the URI-based reference) has additional child elements that are not represented in the definition of the `EcucDestinationUriPolicy`.⌋

[constr_3120] `EcucDestinationUriPolicy.container` always required

Imposition time: CP: `IT_EcucDef`

⌈`EcucDestinationUriPolicy.container` shall exist exactly once.⌋

Example 2.1 shows the definition of the destinationUri "/Example/UriSetA/Uri1". The `EcucDestinationUriPolicy` qualifies the targetContainer with the shortName "UriReferableContainer" and one parameter "InterestingParam1" of type `EcucIntegerParamDef`. In Example 2.2, the module "UriTarget" defines a fitting container and the module "UriRef" defines an according `EcucUriReferenceDef`.

Example 2.1

```
<AR-PACKAGE>
  <SHORT-NAME>EcucDestinationUriDefSet</SHORT-NAME>
  <ELEMENTS>
    <ECUC-DESTINATION-URI-DEF-SET>
      <SHORT-NAME>UriSetA</SHORT-NAME>
      <DESTINATION-URI-DEFS>
        <ECUC-DESTINATION-URI-DEF>
          <SHORT-NAME>Uri1</SHORT-NAME>
          <DESTINATION-URI-POLICY>
            <CONTAINERS>
              <ECUC-PARAM-CONF-CONTAINER-DEF>
                <SHORT-NAME>UriReferableContainer</SHORT-NAME>
                <PARAMETERS>
                  <ECUC-INTEGER-PARAM-DEF>
                    <SHORT-NAME>InterestingParam1</SHORT-NAME>
                    <LOWER-MULTIPLICITY>1</LOWER-MULTIPLICITY>
                    <UPPER-MULTIPLICITY>1</UPPER-MULTIPLICITY>
                    <MAX>255</MAX>
                    <MIN>1</MIN>
                  </ECUC-INTEGER-PARAM-DEF>
                </PARAMETERS>
              </ECUC-PARAM-CONF-CONTAINER-DEF>
            </CONTAINERS>
          </DESTINATION-URI-POLICY>
        </ECUC-DESTINATION-URI-DEF>
      </DESTINATION-URI-DEFS>
    </ECUC-DESTINATION-URI-DEF-SET>
  </ELEMENTS>
</AR-PACKAGE>
```

Example 2.2

```
<AR-PACKAGE>
  <SHORT-NAME>EcucModuleDefs</SHORT-NAME>
  <ELEMENTS>
    <ECUC-MODULE-DEF>
      <SHORT-NAME>UriTarget</SHORT-NAME>
      <CONTAINERS>
        <ECUC-PARAM-CONF-CONTAINER-DEF>
          <SHORT-NAME>UriReferableContainer</SHORT-NAME>
          <DESTINATION-URI-REFS>
            <DESTINATION-URI-REF DEST="ECUC-DESTINATION-URI-DEF">/
              Example/EcucDestinationUriDefSet/UriSetA/Uri1</
              DESTINATION-URI-REF>
          </DESTINATION-URI-REFS>
          <PARAMETERS>
            <ECUC-INTEGER-PARAM-DEF>
              <SHORT-NAME>InterestingParam1</SHORT-NAME>
              <LOWER-MULTIPLICITY>1</LOWER-MULTIPLICITY>
              <UPPER-MULTIPLICITY>1</UPPER-MULTIPLICITY>
              <MAX>255</MAX>
              <MIN>1</MIN>
            </ECUC-INTEGER-PARAM-DEF>
          </PARAMETERS>
        </ECUC-PARAM-CONF-CONTAINER-DEF>
      </CONTAINERS>
    </ECUC-MODULE-DEF>
    <ECUC-MODULE-DEF>
      <SHORT-NAME>UriRef</SHORT-NAME>
      <CONTAINERS>
        <ECUC-PARAM-CONF-CONTAINER-DEF>
          <SHORT-NAME>Container1</SHORT-NAME>
          <REFERENCES>
            <ECUC-URI-REFERENCE-DEF>
              <SHORT-NAME>UriRef_Uri1</SHORT-NAME>
              <DESTINATION-URI-REF DEST="ECUC-DESTINATION-URI-DEF">
                /Example/EcucDestinationUriDefSet/UriSetA/Uri1</
                DESTINATION-URI-REF>
            </ECUC-URI-REFERENCE-DEF>
          </REFERENCES>
        </ECUC-PARAM-CONF-CONTAINER-DEF>
      </CONTAINERS>
    </ECUC-MODULE-DEF>
  </ELEMENTS>
</AR-PACKAGE>
```

2.3.7 Derived Parameter Specification

The parameter definitions introduced in the previous sections are meant to define configuration parameter types regardless how the actual values will be captured. But since the ECU Configuration is dependent on lots of other input information many values for

the configuration of the BSW and the RTE can be taken over or calculated from other values already available in the description (e.g. the System Extract or the Software-Component description) or other sections of the ECU Configuration. Such configuration parameters are called Derived Configuration Parameters.

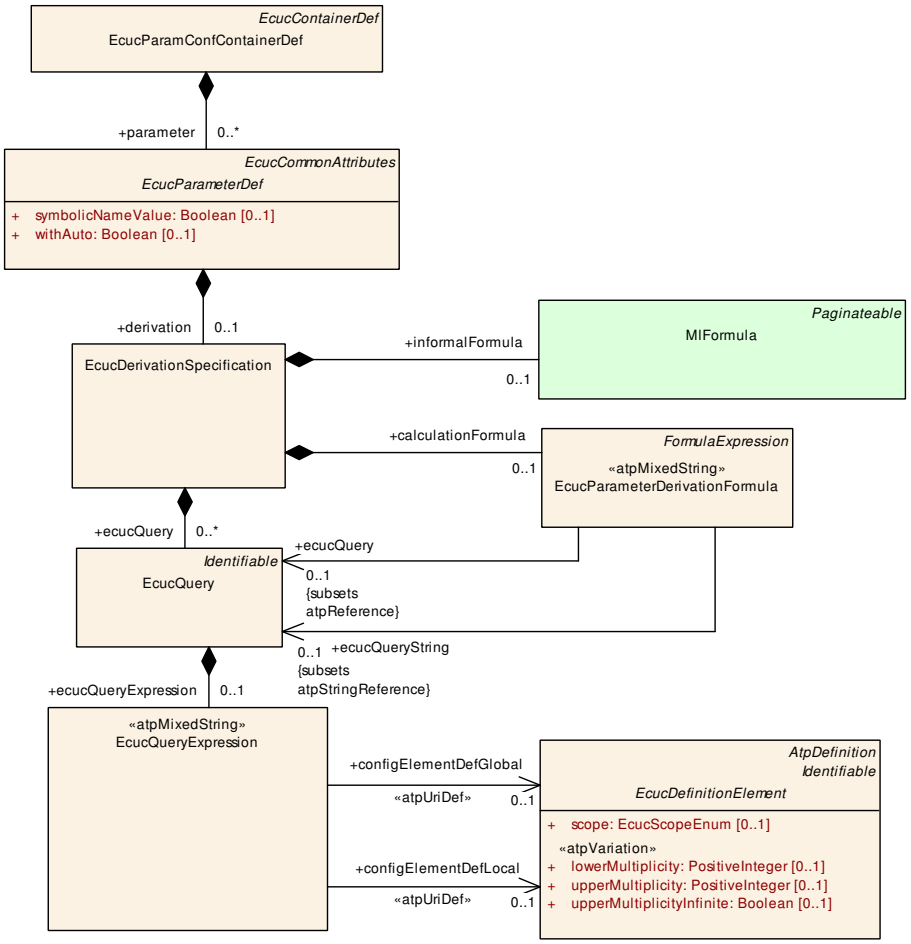

Figure 2.22: Definition of Derived Parameters

[constr_3584] `EcucQuery.ecucQueryExpression` always required

Imposition time: CP: `IT_EcucDef`

⌈The attribute `EcucQuery.ecucQueryExpression` shall always be defined.⌋

Class	**EcucDerivationSpecification**
Note	Allows to define configuration items that are calculated based on the value of • other parameter values • elements (attributes/classes) defined in other AUTOSAR templates such as System template and SW component template This Class is only used by the AUTOSAR Classic Platform.
Base	*ARObject*
Aggregated by	*EcucParameterDef*.derivation

▽

△

Class	**EcucDerivationSpecification**			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
calculation Formula	EcucParameter DerivationFormula	0..1	aggr	Definition of the formula used to calculate the value of the configuration element. **Tags:** atp.Splitkey=calculationFormula
ecucQuery	EcucQuery	*	aggr	Query to the ECU Configuration Description. **Tags:** atp.Splitkey=ecucQuery, ecucQuery.shortName
informalFormula	MIFormula	0..1	aggr	Informal description of the derivation used to calculate the value of the configuration element. **Tags:** atp.Splitkey=informalFormula

Table 2.32: EcucDerivationSpecification

[TPS_ECUC_02047] Derivation of parameter values ⌈ For each EcucParameterDef it can be specified how the parameter value will be computed. This is captured in the element EcucDerivationSpecification.⌋

[TPS_ECUC_02129] Informal description of the derivation ⌈ For all EcucParameterDef types an informal description of the derivation can be specified in the element informalFormula.⌋

[TPS_ECUC_02128] Formal description of the derivation ⌈ For the EcucParameterDef types

- EcucBooleanParamDef
- EcucIntegerParamDef
- EcucFloatParamDef

a formal calculationFormula can be specified in the element EcucParameterDerivationFormula.⌋

Note: The application of the formal calculation formula to the above mentioned types is due to the fact that the result of the calculation formula is numerical.

2.3.7.1 Derived Parameter Calculation Formula

A derivation of a Configuration Parameter value can be specified by an informal Calculation Formula or by a formal language that can be used to specify the computational rules (see figure 2.22). The formal language is defined in the Generic Structure Template [4]. With this formal language it is possible to express dependencies between parameters and e.g. to calculate a value of one parameter based on other parameter values.

Class	«atpMixedString» **EcucParameterDerivationFormula**			
Note	This formula is intended to specify how an ecu parameter can be derived from other information in the Autosar Templates. This Class is only used by the AUTOSAR Classic Platform.			
Base	*ARObject*, *FormulaExpression*			
Aggregated by	EcucDerivationSpecification.calculationFormula			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
ecucQuery	EcucQuery	0..1	ref	This is one particular EcucQuery used in the calculation formula.
ecucQuery String	EcucQuery	0..1	ref	This indicates that the referenced query shall return a string.

Table 2.33: EcucParameterDerivationFormula

The informal Calculation Formula (`MlFormula`) can be used for the same purpose. But here, the rules how the derived values are computed are not defined. Different representations can be used to specify such an informal computational rule. More details can be found in MSRSW. Although the `MlFormula` is informal there can be some programming language syntax and semantics interpreted.

To derive Configuration Parameter values with the formal calculation formula one or several `EcucQuery`s can be defined. An `EcucQuery` is `Identifiable` and aggregates one `EcucQueryExpression`. The `EcucQueryExpression` defines a query to the `ECU Configuration Values` and outputs the result as a numerical value. Four functions are currently supported by the `EcucQueryExpression`: *count*, *value*, *deref* and *refvalue*. Due the `atpMixedString` nature of the `EcucQueryExpression` several function keywords mixed with several local and global references[11] can be defined within an `EcucQueryExpression`.

Class	**EcucQuery**			
Note	Defines a query to the ECUC Description. This Class is only used by the AUTOSAR Classic Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	EcucConditionSpecification.ecucQuery, EcucDerivationSpecification.ecucQuery, EcucValidation Condition.ecucQuery			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
ecucQuery Expression	EcucQueryExpression	0..1	aggr	This is the EcucQuery used in the calculation formula or the condition formula. **Tags:** atp.Splitkey=ecucQueryExpression

Table 2.34: EcucQuery

Class	«atpMixedString» **EcucQueryExpression**
Note	Defines a query expression to the ECUC Description and output the result as an numerical value. Due to the "mixedString" nature of the formula there can be several EcuQueryExpressions used. This Class is only used by the AUTOSAR Classic Platform.
Base	*ARObject*

▽

[11] `configElementDefLocal`, `configElementDefGlobal`

△

Class	«atpMixedString» **EcucQueryExpression**			
Aggregated by	EcucQuery.ecucQueryExpression			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
configElement DefGlobal	EcucDefinitionElement	0..1	ref	The EcucQueryExpression points to an EcucDefinition Element that is used to find an element in the Ecuc Description. In order to find the right element in the Ecuc Description a search is necessary. If the complete Ecuc Description needs to be searched this global reference shall be used. Due to the "mixedString" nature of the EcucQueryExpression several references to Ecuc DefintionElements can be used in one EcucQuery Expression. **Stereotypes:** atpUriDef
configElement DefLocal	EcucDefinitionElement	0..1	ref	The EcucQueryExpression points to an EcucDefinition Element that is used to find an element in the Ecuc Description. In order to find the right element in the Ecuc Description a search is necessary. If the search is executed inside of the same module that contains the EcucQuery this local reference shall be used. Due to the "mixedString" nature of the EcucQueryExpression several references to EcucDefintionElements can be used in one EcucQueryExpression. **Stereotypes:** atpUriDef

Table 2.35: EcucQueryExpression

[constr_5505] Configuration class of the elements of the `EcucQueryExpression`

Imposition time: CP: `IT_EcucDef`

⌈The elements of the `EcucQueryExpression` involved in one calculation formula shall have lower or equal configuration class (where `PreCompile` configuration class is considered to be the lowest and `PostBuild` the highest) with respect to the context element in which the calculation is performed (e.g. a `Link` configuration parameter can not calculate its value based on a `PostBuild` parameters value).⌋

[TPS_ECUC_06018] Input and Output of the *refvalue* function ⌈The *refvalue* function is provided with a `EcucDefinitionElement` and delivers a set of elements from the `ECU Configuration Values` which share the `definition` role of the provided `EcucDefinitionElement`.⌋

[TPS_ECUC_06019] Output of the refvalue function if the EcucDefinitionElement points to a not existing element in the ECU Configuration Parameter Definition ⌈The *refvalue* function shall result in an error if the `EcucDefinitionElement` points to a not existing element in the ECU Configuration Parameter Definition.⌋

[TPS_ECUC_06020] Output of the refvalue function if no element in the `ECU Configuration Values` is found ⌈The *refvalue* function shall return an empty set, if the `EcucDefinitionElement` points to an existing element in the `ECU Configuration Parameter Definition`, but no element in the `ECU Configuration Values` has been found.⌋

[TPS_ECUC_06021] Input and Output of the *deref* function ⌈The *deref* function takes two parameters

1. result of another *deref* function or *refvalue* function, which is an element set
2. reference to a member of the first parameter

and returns the member of the first parameter that is denoted by the second parameter.⌋

[TPS_ECUC_06022] Output of the *deref* function in case the first input parameter is a reference ⌈In case the member of the first parameter is a reference the *deref* function returns the referenced element as a set.⌋

[TPS_ECUC_06023] Cases where the *deref* function reports an error ⌈The *deref* function shall result in an error if

- the first parameter is an empty set
- the first parameter is a set with more than 1 elements[12]
- the first parameter contains one element which is a value (e.g. 5)
- second parameter points to a not existing element in the ECU Configuration Parameter Definition or to the AUTOSAR Schema.

⌋

[TPS_ECUC_06024] Input of the *value* function ⌈The *value* function takes the result of a *deref* function or *refvalue* function, which is an element set.⌋

[TPS_ECUC_06025] Output of the *value* function ⌈The *value* function returns the parameter's value as numerical value.⌋

[TPS_ECUC_06026] Cases where the *value* function reports an error ⌈The *value* function shall result in an error if

- the parameter is an empty set
- the parameter is a set with more than 1 elements[13]
- the parameter's single element does not have a value (e.g. is a container)

⌋

[TPS_ECUC_06057] Input of the *strValue* function ⌈The *strValue* function takes the result of a *deref* function or *refvalue* function, which is an element set.⌋

[TPS_ECUC_06058] Output of the *strValue* function ⌈The *strValue* function returns the parameter's value as string.⌋

[12] The *deref* function shall only be applied to element sets which are guaranteed to contain only up to 1 element.

[13] The *value* function shall only be applied to element sets which are guaranteed to contain only up to 1 element.

[TPS_ECUC_06059] Cases where the *strValue* function reports an error ⌈The *strValue* function shall result in an error if

- the parameter is an empty set
- the parameter is a set with more than 1 elements[14]
- the parameter's single element does not have a value (e.g. is a container)

⌋

[TPS_ECUC_06060] Input of the *valueAt* function ⌈The *valueAt* function takes the result of a *deref* function or *refvalue* function, which is an element set and a zero-based position argument.⌋

[TPS_ECUC_06061] Output of the *valueAt* function ⌈The *valueAt* function returns the value of the parameter as numerical value at the position according to the sorting criteria defined in section xxx⌋

[TPS_ECUC_06062] Cases where the *valueAt* function reports an error ⌈The *valueAt* function function shall result in an error if

- the parameter is an empty set
- the parameter is a set with more than 1 elements
- the parameter's single element does not have a value (e.g. is a container)
- the position is larger than the count-1

⌋

[TPS_ECUC_06063] Input of the *strValueAt* function ⌈The *strValueAt* function takes the result of a *deref* function or *refvalue* function, which is an element set and a zero-based position argument.⌋

[TPS_ECUC_06064] Output of the *strValueAt* function ⌈The *strValueAt* function returns the value of the parameter as string at the position according to the sorting criteria defined in section x.x.x.⌋

[TPS_ECUC_06065] Cases where the *strValueAt* function reports an error ⌈The *strValueAt* function function shall result in an error if

- the parameter is an empty set
- the parameter is a set with more than 1 elements
- the parameter's single element does not have a value (e.g. is a container)
- the position is larger than the count-1

[14] The *strValue* function shall only be applied to element sets which are guaranteed to contain only up to 1 element.

⌋

[TPS_ECUC_06027] Input of the *count* function ⌈The *count* function gets the result of the *deref* or *refvalue* function as input parameter.⌋

[TPS_ECUC_06028] Output of the *count* function ⌈The *count* function returns the number of elements in the input parameter set.⌋

[TPS_ECUC_06029] Output of the *count* function in case the input parameter set is empty ⌈The *count* function returns zero if the input parameter set is empty.⌋

In order to find the referenced element in the ECU Configuration Values, the reference to the EcucDefinitionElement needs to be traced. If the complete ECU Configuration Values needs to be searched, a global reference (configElementDefGlobal) shall be used. If the search is executed inside of the same module, a local reference (configElementDefLocal) is sufficient.

The following section shows the EcucQueryExpression syntax:

```
ecuQueryExpr : (valueExpr|stringValueExpr|valueAtExpr|stringValueAtExpr|countExpr);
valueExpr : 'value('(derefExpr | refValueExpr) ')';
stringValueExpr : 'strValue('(derefExpr | refValueExpr) ')';
valueAtExpr : 'valueAt('(derefExpr | refValueExpr) ',' index ')'
stringValueAtExpr : 'strValueAt('(derefExpr | refValueExpr) ',' index ')'
countExpr : 'count('(derefExpr | refValueExpr) ')';
refValueExpr : 'refvalue(' refExpr ')';
derefExpr : 'deref('(derefExpr| refValueExpr) ',' refString ')';
refExpr : (localRef | globalRef);
localRef : '<CONFIG-ELEMENT-DEF-LOCAL-REF DEST="' NCName* '">'
            refString '</CONFIG-ELEMENT-DEF-LOCAL-REF>';
globalRef : '<CONFIG-ELEMENT-DEF-GLOBAL-REF DEST="' NCName* '">'
            refString '</CONFIG-ELEMENT-DEF-GLOBAL-REF>';
refString : '/'NCName('/'NCName)*;
index: '0' | ('1'..'9')('0'..'9')*;
NCName : (Letter) (Letter | ('0'..'9') | '_')*;
```

Figure 2.23 shows a COM Gateway example where the CheckConsistency boolean parameter is calculated. This parameter checks the length of the Source Signal and compares it with the length of the Destination Signal. If the length of both signals is equal this parameter is set to true, otherwise to false. An XML extract from an ECUC Parameter Definition file is is shown in Listing 2.21.

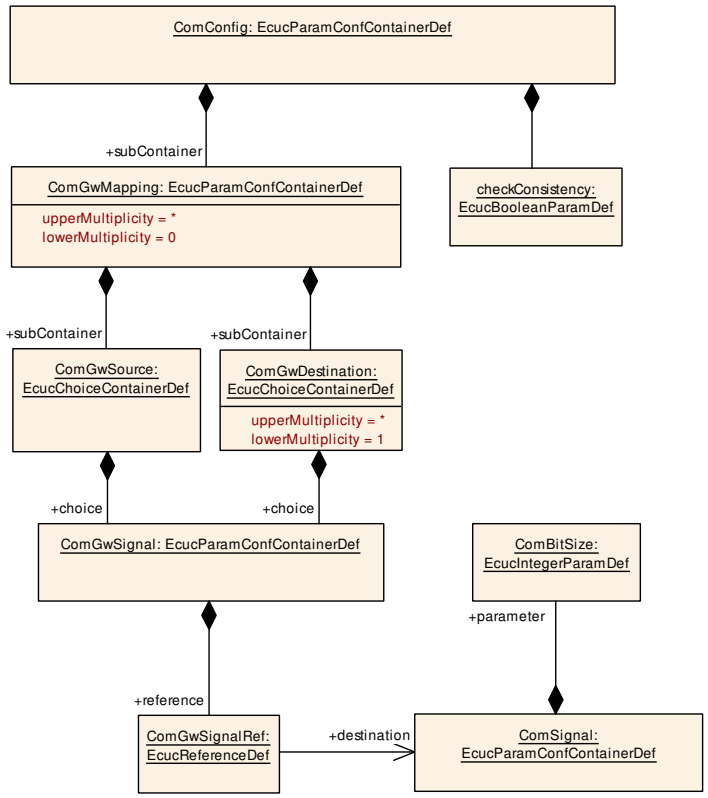

Figure 2.23: Calculation Formula Example

To determine the parameter value the `EcucDerivationSpecification` within the `CheckConsistency` parameter aggregates two `EcucQueries`.

The first `EcucQuery` "getSourceSignalLength" contains a `EcucQueryExpression` with a local reference to the `ComGwSignalRef` element. To get the signal length from the referenced `ComGwSignal` two *deref* functions are used. The first *deref* function takes the reference to the `ComGwSignalRef` element as input and returns the `ComGwSignal` that is searched by the second input parameter. The second *deref* function takes the `ComGwSignal` as the first input parameter and the reference to the searched ECUC parameter within the `ComGwSignal` as the second input parameter and returns the `ComBitSize` parameter. The value of the `ComBitSize` parameter is provided by the *value* function.

To find the right source signal in the `ECU Configuration Values` the biggest common prefix from the local reference and from the `CheckConsistency` parameter path is used as entry point to the `ECU Configuration Values`. In this example the biggest common prefix is the following path: /AUTOSAR/EcucDefs/Com/ComConfig/-ComGwMapping/.

The second `EcucQuery` "getDestinationSignalLength" provides the `ComBitSize` Parameter Value of the destination Signal accordingly.

The `CalculationFormula` compares both values and determines the value for the `CheckConsistency` parameter. The corresponding `ECU Configuration Values ARXML` file extract is shown in Listing 2.21.

```
<ECUC-MODULE-DEF>
  <SHORT-NAME>Com</SHORT-NAME>
  <CONTAINERS>
    <ECUC-PARAM-CONF-CONTAINER-DEF>
      <SHORT-NAME>ComConfig</SHORT-NAME>
      <SUB-CONTAINERS>
        <ECUC-PARAM-CONF-CONTAINER-DEF>
          <SHORT-NAME>ComGwMapping</SHORT-NAME>
          <PARAMETERS>
            <ECUC-BOOLEAN-PARAM-DEF>
              <SHORT-NAME>CheckConsistency</SHORT-NAME>
              <DERIVATION>
                <CALCULATION-FORMULA>
(<ECUC-QUERY-REF DEST="ECUC-QUERY">/AUTOSAR/Com/ComConfig/ComGwMapping/
    CheckConsistency/getSourceSignalLength</ECUC-QUERY-REF> ==
                  <ECUC-QUERY-REF DEST="ECUC-QUERY">/AUTOSAR/Com/ComConfig/
                     ComGwMapping/CheckConsistency/
                     getDestinationSignalLength</ECUC-QUERY-REF>)
                </CALCULATION-FORMULA>
                <ECUC-QUERYS>
                  <ECUC-QUERY>
                    <SHORT-NAME>getSourceSignalLength</SHORT-NAME>
                     <ECUC-QUERY-EXPRESSION>
value(
deref(
deref(
refvalue(<CONFIG-ELEMENT-DEF-LOCAL-REF DEST="ECUC-CHOICE-REFERENCE-DEF">/
    AUTOSAR/EcucDefs/Com/ComConfig/ComGwMapping/ComGwSource/ComGwSignal/
    ComGwSignalRef</CONFIG-ELEMENT-DEF-LOCAL-REF>),
/AUTOSAR/EcucDefs/Com/ComConfig/ComGwMapping/ComGwSource/ComGwSignal/
    ComGwSignalRef),
/ComBitSize)
)
                    </ECUC-QUERY-EXPRESSION>
                  </ECUC-QUERY>
                  <ECUC-QUERY>
                    <SHORT-NAME>getDestinationSignalLength</SHORT-NAME>
                    <ECUC-QUERY-EXPRESSION>
value(
deref(
deref(
refvalue(<CONFIG-ELEMENT-DEF-LOCAL-REF DEST="ECUC-CHOICE-REFERENCE-DEF">/
    AUTOSAR/EcucDefs/Com/ComConfig/ComGwMapping/ComGwDestination/ComGwSignal
    /ComGwSignalRef</CONFIG-ELEMENT-DEF-LOCAL-REF>),
/AUTOSAR/EcucDefs/Com/ComConfig/ComGwMapping/ComGwDestination/ComGwSignal/
    ComGwSignalRef),
/ComBitSize)
)
                    </ECUC-QUERY-EXPRESSION>
                  </ECUC-QUERY>
                </ECUC-QUERYS>
              </DERIVATION>
            </ECUC-BOOLEAN-PARAM-DEF>
          </PARAMETERS>
        </ECUC-PARAM-CONF-CONTAINER-DEF>
      </SUB-CONTAINERS>
```

```
    </ECUC-PARAM-CONF-CONTAINER-DEF>
  </CONTAINERS>
</ECUC-MODULE-DEF>
```

Listing 2.21: `calculationFormula` comparing values

The next Listing 2.22 shows the usage of the *count* operation. Within the COM module an Integer Parameter `countNoOfCanDrv` is introduced which counts the available CanDrv modules. To cover all CanDrv modules a global reference is used.

```
<ECUC-MODULE-DEF>
  <SHORT-NAME>Com</SHORT-NAME>
  <CONTAINERS>
    <ECUC-PARAM-CONF-CONTAINER-DEF>
      <SHORT-NAME>ComConfig</SHORT-NAME>
      <PARAMETERS>
        <ECUC-INTEGER-PARAM-DEF>
          <SHORT-NAME>numberOfCanDrivers</SHORT-NAME>
          <DERIVATION>
            <CALCULATION-FORMULA>
              <ECUC-QUERY-REF DEST="ECUC-QUERY">/AUTOSAR/Com/ComConfig/
                  numberOfCanDrivers/countNoOfCanDrv</ECUC-QUERY-REF>
            </CALCULATION-FORMULA>
            <ECUC-QUERYS>
              <ECUC-QUERY>
                <SHORT-NAME>countNoOfCanDrv</SHORT-NAME>
                <ECUC-QUERY-EXPRESSION>
count(
refvalue(<CONFIG-ELEMENT-DEF-GLOBAL-REF DEST="ECUC-MODULE-DEF">/AUTOSAR/
    EcucDefs/Can</CONFIG-ELEMENT-DEF-GLOBAL-REF>)
)
                </ECUC-QUERY-EXPRESSION>
              </ECUC-QUERY>
            </ECUC-QUERYS>
          </DERIVATION>
        </ECUC-INTEGER-PARAM-DEF>
      </PARAMETERS>
    </ECUC-PARAM-CONF-CONTAINER-DEF>
  </CONTAINERS>
</ECUC-MODULE-DEF>
```

Listing 2.22: `calculationFormula` using count

A third example 2.24 shows a reference into the System Description. The referenced `ComSignal` contains a `ForeignReference` into the System Template (SystemTemplateSystemSignalRef). The searched `startPosition` attribute is defined in the System Template and describes a bitposition of a `SystemSignal` within a `PDU`.

To get the value of this attribute three *deref* functions are used. The first *deref* function provides the `ComSignal`. The second *deref* function provides the `ISignalToPduMapping` element of the System Description and the third *deref* function returns the `startPosition` attribute of the `ISignalToPduMapping` element. The attribute value is provided by the *value* function and is used in the calculation formula.

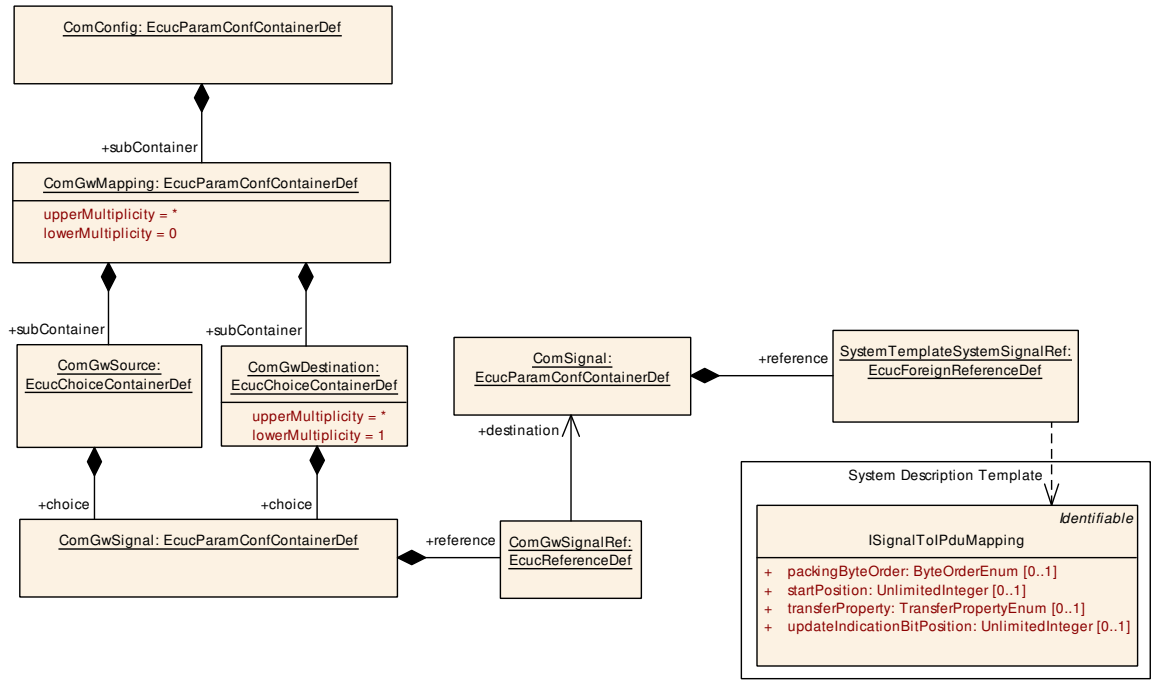

Figure 2.24: Calculation Formula Example

```
<ECUC-MODULE-DEF>
  <SHORT-NAME>Com</SHORT-NAME>
  <CONTAINERS>
    <ECUC-PARAM-CONF-CONTAINER-DEF>
      <SHORT-NAME>ComConfig</SHORT-NAME>
      <SUB-CONTAINERS>
        <ECUC-PARAM-CONF-CONTAINER-DEF>
          <SHORT-NAME>ComGwMapping</SHORT-NAME>
          <PARAMETERS>
            <ECUC-INTEGER-PARAM-DEF>
              <SHORT-NAME>startPositionBits</SHORT-NAME>
              <DERIVATION>
                <CALCULATION-FORMULA>
                  <ECUC-QUERY-REF DEST="ECUC-QUERY">/AUTOSAR/Com/ComConfig/
                      ComGwMapping/startPositionBits/
                      getSourceSignalStartPosition</ECUC-QUERY-REF>* 8
                </CALCULATION-FORMULA>
                <ECUC-QUERYS>
                  <ECUC-QUERY>
                    <SHORT-NAME>getSourceSignalStartPosition</SHORT-NAME>
                    <ECUC-QUERY-EXPRESSION>
value(
deref(
deref(
deref(
refvalue(<CONFIG-ELEMENT-DEF-LOCAL-REF DEST="ECUC-CHOICE-REFERENCE-DEF">/
    AUTOSAR/EcucDefs/Com/ComConfig/ComGwMapping/ComGwSource/ComGwSignal/
    ComGwSignalRef</CONFIG-ELEMENT-DEF-LOCAL-REF>),
/AUTOSAR/EcucDefs/Com/ComConfig/ComGwMapping/ComGwSource/ComGwSignal/
    ComGwSignalRef),
/SystemTemplateSystemSignalRef),
/SystemTemplateSystemSignalRef),
```

```
/startPosition)
)
                    </ECUC-QUERY-EXPRESSION>
                  </ECUC-QUERY>
                </ECUC-QUERYS>
              </DERIVATION>
            </ECUC-INTEGER-PARAM-DEF>
          </PARAMETERS>
        </ECUC-PARAM-CONF-CONTAINER-DEF>
      </SUB-CONTAINERS>
    </ECUC-PARAM-CONF-CONTAINER-DEF>
  </CONTAINERS>
</ECUC-MODULE-DEF>
```

Listing 2.23: `calculationFormula` reference to System Description

2.3.7.2 Restrictions on Configuration Class of Derived Parameters

Derived Parameters have to be defined similar to plain configuration parameters which means that also the configuration class has to be specified in the actual implementation of the configuration. But since derived parameters do depend on other information there are certain restrictions applicable which reduce the degree of freedom what kind of configuration class a derived parameter might be.

If the derived parameter is derived from other Configuration Parameters in the `ECU Configuration Values` then certain rules have to be applied:

- **[TPS_ECUC_02058] Derivation of information from `PreCompile` parameters** ⌈ If the derived parameter uses information from parameters defined as `PreCompile`, then the derived parameter can be of any configuration class.⌋
- **[TPS_ECUC_02056] Derivation of information from `Link` parameters** ⌈ If the derived parameter uses information from parameters defined as `Link`, then the derived parameter shall be of either `Link` or `PostBuild` configuration class.⌋
- **[TPS_ECUC_02057] Derivation of information from `PostBuild` parameters** ⌈ If the derived parameter uses information from parameters defined as `PostBuild`, then the derived parameter shall be of `PostBuild` configuration class.⌋
- **[TPS_ECUC_08017] Derivation of information from parameter values bound at `PreCompile` time** ⌈ If the derived parameter uses information from parameter values which are bound at `PreCompile` time, then the derived parameter value can be bound at any time.⌋
- **[TPS_ECUC_08018] Derivation of information from parameter values bound at `Link` time** ⌈ If the derived parameter uses information from parameter values which are bound at `Link` time, then the derived parameter value shall be bound at either `Link` or `PostBuild` time.⌋
- **[TPS_ECUC_08019] Derivation of information from parameter values bound at `PostBuild` time** ⌈ If the derived parameter uses information from parameter values which are bound at `PostBuild` time, then the derived parameter value shall be bound at `PostBuild` time.⌋

2.3.8 Existence dependence of ECUC Parameter Definition elements

ECUC Parameter Values can be calculated from other parameter values that are available in other sections of the ECU Configuration. Such derived configuration parameters are described in detail in chapter 2.3.7. But also the existence of a ECUC Container, Parameter and Reference definition elements can depend on the setting of ECUC Parameter Values. Such it is for example possible to define parameters that are only

considered if a specific switch parameter is set to a certain value. Otherwise these parameters are ignored.

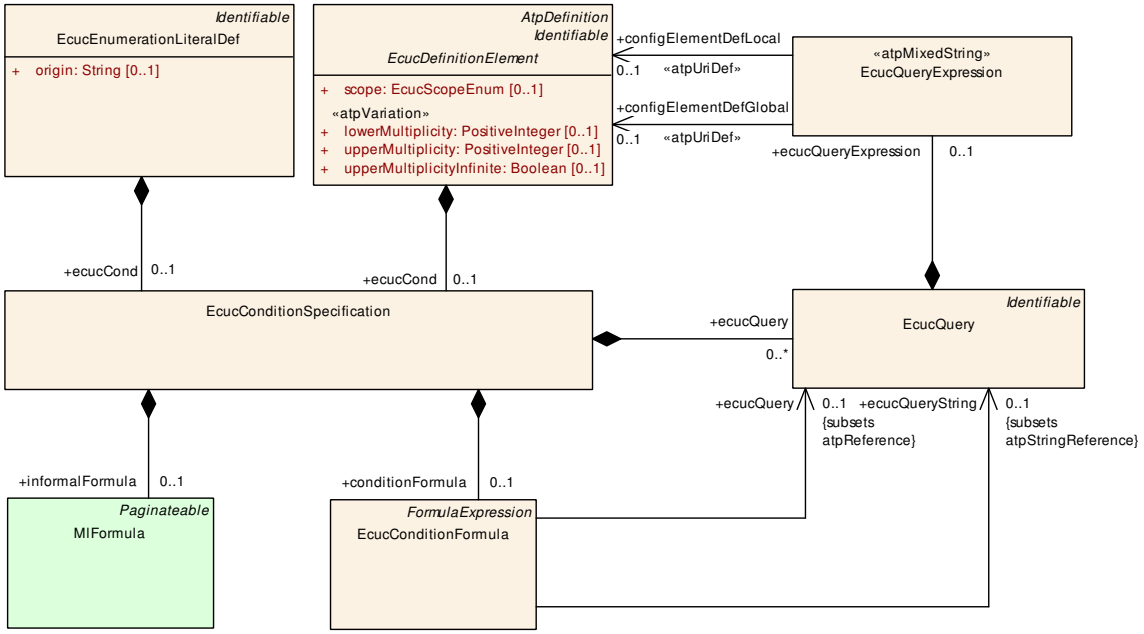

Figure 2.25: Existence dependence of parameter definitions and literal definitions

[constr_3585] `EcucConditionFormula.ecucQuery` always required

Imposition time: CP: `IT_EcucDef`

⌈The attribute `EcucConditionFormula.ecucQuery` shall always be defined.⌋

[constr_3586] `EcucConditionFormula.ecucQueryString` always required

Imposition time: CP: `IT_EcucDef`

⌈The attribute `EcucConditionFormula.ecucQueryString` shall always be defined.⌋

To allow the description of such existence dependencies the `EcucDefinitionElement` and the `EcucEnumerationLiteralDef` aggregate the `EcucConditionSpecification`. The `EcucConditionSpecification` aggregates an `EcucConditionFormula` or a informal Calculation Formula (`MlFormula`). If the `EcucConditionFormula` evaluates to true the parameter definition/literal definition shall be processed as specified. Otherwise the parameter definition/literal definition shall be ignored. The informal Calculation Formula (`MlFormula`) can be used for the same purpose. But here, the rules how the condition is evaluated are not defined.

An `EcucQuery` to the `ECU Configuration Values` serves as an argument for the `EcucConditionFormula`. Due the `atpMixedString` nature of the `EcucConditionFormula` several `EcucQueries` can be defined within an `EcucConditionFormula`.

An EcucQuery is Identifiable and aggregates one EcucQueryExpression. The EcucQueryExpression outputs the result as a numerical value. The EcucQueryExpression syntax is described in chapter 2.3.7.1.

Class	EcucConditionSpecification			
Note	Allows to define existence dependencies based on the value of parameter values. This Class is only used by the AUTOSAR Classic Platform.			
Base	ARObject			
Aggregated by	EcucDefinitionElement.ecucCond, EcucEnumerationLiteralDef.ecucCond			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
conditionFormula	EcucConditionFormula	0..1	aggr	Definition of the formula used to define existence dependencies. **Tags:** atp.Splitkey=conditionFormula
ecucQuery	EcucQuery	*	aggr	Query to the ECU Configuration Description. **Tags:** atp.Splitkey=ecucQuery, ecucQuery.shortName
informalFormula	MIFormula	0..1	aggr	Informal description of the condition used to to define existence dependencies. **Tags:** atp.Splitkey=informalFormula

Table 2.36: EcucConditionSpecification

Class	«atpMixedString» EcucConditionFormula			
Note	This formula shall yield a boolean expression depending on ecuc queries. Note that the EcucCondition Formula is a mixed string. Therefore, the properties have the upper multiplicity 1. This Class is only used by the AUTOSAR Classic Platform.			
Base	ARObject, FormulaExpression			
Aggregated by	EcucConditionSpecification.conditionFormula, EcucValidationCondition.validationFormula			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
ecucQuery	EcucQuery	0..1	ref	The EcucQuery serves as a argument for the formula.
ecucQueryString	EcucQuery	0..1	ref	This indicates that the referenced query shall return a string.

Table 2.37: EcucConditionFormula

As an example, let's say there is a module ExampleModule with a feature ExampleFeature. This feature is has some settings in a container ExampleFeatureGeneral. There is a parameter to enable or disable the ExampleFeature, called EnableExampleFeature. If the value of EnableExampleFeature is set to true, the container ExampleFeatureGeneral and its content shall be available for configuration. If the value of EnableExampleFeature is set to false the container ExampleFeatureGeneral shall not be considered for configuration.

```
<ECUC-MODULE-DEF>
  <SHORT-NAME>ExampleModule</SHORT-NAME>
  <LOWER-MULTIPLICITY>0</LOWER-MULTIPLICITY>
  <UPPER-MULTIPLICITY>1</UPPER-MULTIPLICITY>
  <CONTAINERS>
    <ECUC-PARAM-CONF-CONTAINER-DEF>
      <SHORT-NAME>SomeContainer</SHORT-NAME>
      <LOWER-MULTIPLICITY>1</LOWER-MULTIPLICITY>
      <UPPER-MULTIPLICITY>1</UPPER-MULTIPLICITY>
      <PARAMETERS>
```

```
        <ECUC-BOOLEAN-PARAM-DEF>
          <SHORT-NAME>SomeParameter</SHORT-NAME>
          <LOWER-MULTIPLICITY>1</LOWER-MULTIPLICITY>
          <UPPER-MULTIPLICITY>1</UPPER-MULTIPLICITY>
          <!-- ... -->
        </ECUC-BOOLEAN-PARAM-DEF>
        <ECUC-BOOLEAN-PARAM-DEF>
          <SHORT-NAME>EnableExampleFeature</SHORT-NAME>
          <LOWER-MULTIPLICITY>1</LOWER-MULTIPLICITY>
          <UPPER-MULTIPLICITY>1</UPPER-MULTIPLICITY>
          <DEFAULT-VALUE>false</DEFAULT-VALUE>
        </ECUC-BOOLEAN-PARAM-DEF>
      </PARAMETERS>
      <SUB-CONTAINERS>
        <ECUC-PARAM-CONF-CONTAINER-DEF>
          <SHORT-NAME>ExampleFeatureGeneral</SHORT-NAME>
          <ECUC-COND>
            <CONDITION-FORMULA>
              <ECUC-QUERY-REF DEST="ECUC-QUERY">/AUTOSAR/ExampleModule/
                  SomeContainer/ExampleFeatureGeneral/
                  GetExampleFeatureEnabled</ECUC-QUERY-REF>
            </CONDITION-FORMULA>
            <ECUC-QUERYS>
              <ECUC-QUERY>
                <SHORT-NAME>GetExampleFeatureEnabled</SHORT-NAME>
                <ECUC-QUERY-EXPRESSION>
value(
refvalue(<CONFIG-ELEMENT-DEF-LOCAL-REF DEST="ECUC-BOOLEAN-PARAM-DEF">/
    AUTOSAR/EcucDefs/ExampleModule/SomeContainer/EnableExampleFeature</
    CONFIG-ELEMENT-DEF-LOCAL-REF>)
)
                </ECUC-QUERY-EXPRESSION>
              </ECUC-QUERY>
            </ECUC-QUERYS>
          </ECUC-COND>
          <LOWER-MULTIPLICITY>0</LOWER-MULTIPLICITY>
          <UPPER-MULTIPLICITY>1</UPPER-MULTIPLICITY>
          <PARAMETERS>
            <!-- ... -->
          </PARAMETERS>
        </ECUC-PARAM-CONF-CONTAINER-DEF>
      </SUB-CONTAINERS>
    </ECUC-PARAM-CONF-CONTAINER-DEF>
  </CONTAINERS>
</ECUC-MODULE-DEF>
```

Listing 2.24: `EcucConditionFormula` Example

The condition formula is part of the `ExampleFeatureGeneral` container definition (see Listing 2.24). The formula itself is pretty simple, it just returns the value of the `EcucQuery` with the name `GetExampleFeatureEnabled`.

The `EcucQuery` looks for an element in the `ECU Configuration Values` which matches the definition
(`/AUTOSAR/EcucDefs/ExampleModule/SomeContainer/EnableExample-`

Feature)
in the local context using the refvalue function.

The EcucQuery then takes the value of the element and returns. Since the element is of boolean type the result of the EcucQuery is already a boolean value which can be processed by the condition formula.

2.3.9 Validation conditions

In order to describe validity constrains on a configuration element the ecucValidationCond can define a set of EcucValidationConditions which can be aggregated by any subclass of EcucDefinitionElement.

[TPS_ECUC_02135] Validation of EcucValidationCondition ⌈An EcucValidationCondition of an EcucDefinitionElement is considered *valid* if the validationFormula of that EcucValidationCondition evaluates to true.⌋

[TPS_ECUC_02136] Validation of multiple EcucValidationConditions ⌈A configuration of an EcucDefinitionElement is considered *valid* if all of the defined ecucValidationConds of that EcucDefinitionElement are *valid*.⌋

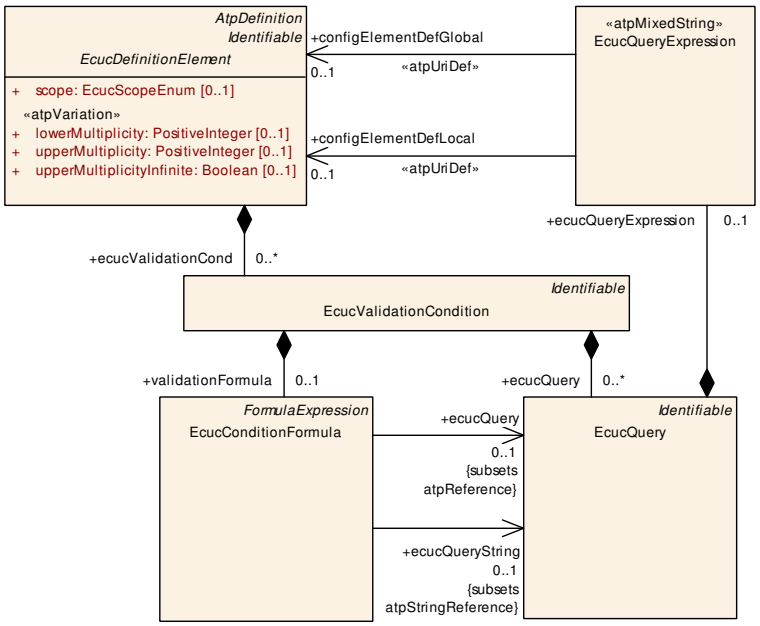

Figure 2.26: Validation condition

[constr_3587] EcucValidationCondition.validationFormula always required

Imposition time: CP: IT_EcucDef

⌈The attribute EcucValidationCondition.validationFormula shall always be defined.⌋

Class	**EcucValidationCondition**			
Note	Validation condition to perform a formula calculation based on EcucQueries. This Class is only used by the AUTOSAR Classic Platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	*EcucDefinitionElement*.ecucValidationCond			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
ecucQuery	EcucQuery	*	aggr	Query to the ECU Configuration Description. **Tags:** atp.Splitkey=ecucQuery, ecucQuery.shortName
validation Formula	EcucConditionFormula	0..1	aggr	Definition of the formula used to define validation condition. **Tags:** atp.Splitkey=validationFormula

Table 2.38: EcucValidationCondition

2.3.10 Multiple aggregation of full container trees that include references

The ECU Configuration Parameter Definitions UML model may define an EcucContainerDef that is aggregated by different EcucParamConfContainerDefs in the role subContainer. In case that the subContainer contains references then some rules apply that are described in the following:

[TPS_ECUC_06089] Multiple aggregation of container trees that include references to other subContainers in the same aggregated container tree ⌈In case an EcucParamConfContainerDef is aggregated by different EcucParamConfContainerDefs in the role subContainer and this aggregated subContainer has an EcucReferenceDef to another subContainer located in the same aggregated EcucParamConfContainerDef structure, then the DESTINATION-REF (in the generated ECU Configuration Parameter Definition XML file) of the EcucReferenceDef shall include the shortName path back to the aggregating EcucParamConfContainerDef in the PATH of the DESTINATION-REF.⌋

The following example explains [TPS_ECUC_06089]: ContainerB and ContainerC are aggregating the same SubContainerA in the ECU Configuration Parameter Definitions UML model:

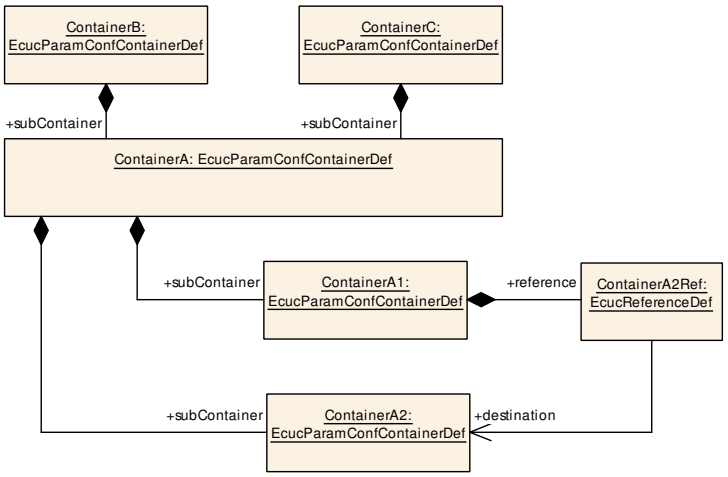

Figure 2.27: Example for multiple aggregation of container trees

The resulting AUTOSAR_MOD_ECUConfigurationParameters.arxml will be defined as follows (simplified). Please note that the PATH of the DESTINATION-REF is different in both ContainerB and ContainerC.

```
<CONTAINERS>
  <ECUC-PARAM-CONF-CONTAINER-DEF>
    <SHORT-NAME>ContainerB</SHORT-NAME>
    <LOWER-MULTIPLICITY>1</LOWER-MULTIPLICITY>
    <UPPER-MULTIPLICITY>1</UPPER-MULTIPLICITY>
    <SUB-CONTAINERS>
      <ECUC-PARAM-CONF-CONTAINER-DEF>
        <SHORT-NAME>ContainerA</SHORT-NAME>
        <LOWER-MULTIPLICITY>1</LOWER-MULTIPLICITY>
        <UPPER-MULTIPLICITY>1</UPPER-MULTIPLICITY>
        <SUB-CONTAINERS>
          <ECUC-PARAM-CONF-CONTAINER-DEF>
            <SHORT-NAME>ContainerA1</SHORT-NAME>
            <LOWER-MULTIPLICITY>1</LOWER-MULTIPLICITY>
            <UPPER-MULTIPLICITY>1</UPPER-MULTIPLICITY>
            <REFERENCES>
              <ECUC-REFERENCE-DEF>
                <SHORT-NAME>ContainerA2Ref</SHORT-NAME>
                <LOWER-MULTIPLICITY>1</LOWER-MULTIPLICITY>
                <UPPER-MULTIPLICITY>1</UPPER-MULTIPLICITY>
                <SCOPE>LOCAL</SCOPE>
                <DESTINATION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF">/
                  AUTOSAR/EcucDefs/ExampleModule/ContainerB/ContainerA/
                  ContainerA2</DESTINATION-REF>
              </ECUC-REFERENCE-DEF>
            </REFERENCES>
          </ECUC-PARAM-CONF-CONTAINER-DEF>
          <ECUC-PARAM-CONF-CONTAINER-DEF>
            <SHORT-NAME>ContainerA2</SHORT-NAME>
            <LOWER-MULTIPLICITY>1</LOWER-MULTIPLICITY>
            <UPPER-MULTIPLICITY>1</UPPER-MULTIPLICITY>
          </ECUC-PARAM-CONF-CONTAINER-DEF>
        </SUB-CONTAINERS>
      </ECUC-PARAM-CONF-CONTAINER-DEF>
```

```
    </SUB-CONTAINERS>
  </ECUC-PARAM-CONF-CONTAINER-DEF>
  <ECUC-PARAM-CONF-CONTAINER-DEF>
    <SHORT-NAME>ContainerC</SHORT-NAME>
    <LOWER-MULTIPLICITY>1</LOWER-MULTIPLICITY>
    <UPPER-MULTIPLICITY>1</UPPER-MULTIPLICITY>
    <SUB-CONTAINERS>
      <ECUC-PARAM-CONF-CONTAINER-DEF>
        <SHORT-NAME>ContainerA</SHORT-NAME>
        <LOWER-MULTIPLICITY>1</LOWER-MULTIPLICITY>
        <UPPER-MULTIPLICITY>1</UPPER-MULTIPLICITY>
        <SUB-CONTAINERS>
          <ECUC-PARAM-CONF-CONTAINER-DEF>
            <SHORT-NAME>ContainerA1</SHORT-NAME>
            <LOWER-MULTIPLICITY>1</LOWER-MULTIPLICITY>
            <UPPER-MULTIPLICITY>1</UPPER-MULTIPLICITY>
            <REFERENCES>
              <ECUC-REFERENCE-DEF>
                <SHORT-NAME>ContainerA2Ref</SHORT-NAME>
                <LOWER-MULTIPLICITY>1</LOWER-MULTIPLICITY>
                <UPPER-MULTIPLICITY>1</UPPER-MULTIPLICITY>
                <DESTINATION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF">/
                    AUTOSAR/EcucDefs/ExampleModule/ContainerC/ContainerA/
                    ContainerA2</DESTINATION-REF>
              </ECUC-REFERENCE-DEF>
            </REFERENCES>
          </ECUC-PARAM-CONF-CONTAINER-DEF>
          <ECUC-PARAM-CONF-CONTAINER-DEF>
            <SHORT-NAME>ContainerA2</SHORT-NAME>
              <LOWER-MULTIPLICITY>1</LOWER-MULTIPLICITY>
              <UPPER-MULTIPLICITY>1</UPPER-MULTIPLICITY>
          </ECUC-PARAM-CONF-CONTAINER-DEF>
        </SUB-CONTAINERS>
      </ECUC-PARAM-CONF-CONTAINER-DEF>
    </SUB-CONTAINERS>
  </ECUC-PARAM-CONF-CONTAINER-DEF>
</CONTAINERS>
```

Listing 2.25: Example for multiple aggregation of container trees in ECU Configuration Parameter Definition ARXML file

[constr_5345] Restriction for a reference destination in case of multiple aggregated EcucParamConfContainerDefs

Imposition time: CP: IT_EcucDef

⌈An EcucReferenceDef or EcucChoiceReferenceDef is not allowed to reference an EcucParamConfContainerDef as destination if

- this EcucParamConfContainerDef is aggregated by several EcucParamConfContainerDefs as subContainer and
- the EcucParamConfContainerDef structures in which the referenced EcucParamConfContainerDef is aggregated are different compared to the Ecuc-

`ParamConfContainerDef` structure in which the `EcucReferenceDef` or `EcucChoiceReferenceDef` is located in.

⌋

2.4 ECU Configuration Value Meta-model

As mentioned in section 2.2 the ECU Configuration Parameter Definition Meta-model provides the means to declare the parameters and their permitted occurrences within a configuration file. This section will specify the complement to that ECU Configuration Parameter Definition on the values side, namely the ECU Configuration Values (as a model) or ECU Configuration Values ARXML (the serialization in the file).

The following sections will depict the ECU Configuration Value Meta-model. Sections 2.4.1 and 2.4.2 will introduce the top-level structure of configuration values and the module configurations, whereas the sections 2.4.3, 2.4.4 and 2.4.5 will describe the means to structure the actual configuration values ARXML file.

2.4.1 ECU Configuration Value Top-Level Structure

The top-level entry point to an AUTOSAR ECU Configuration Values is the EcucValueCollection (see figure 2.28). Because of the inheritance from ARElement the EcucValueCollection can be part of an AUTOSAR description like its counterpart the EcucDefinitionCollection does. Please note that the EcucValueCollection and the EcucDefinitionCollection are independent from each other.

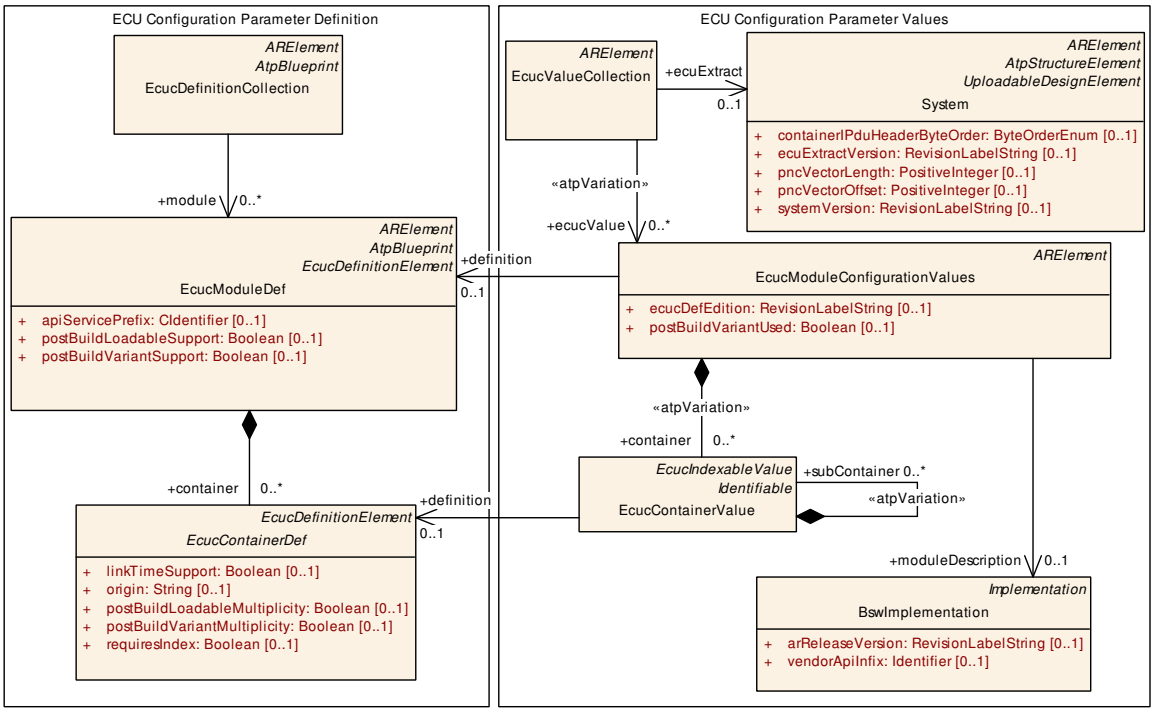

Figure 2.28: ECU Configuration Value Top-Level Structure

[constr_4202] Existence of `EcucValueCollection.ecucValue`

Imposition time: CP: `IT_EcucVal`

⌈`EcucValueCollection.ecucValue` shall exist at least once.⌋

An `EcucValueCollection` without any `EcucModuleConfigurationValues` would not have any effect on the Ecu configuration.

[constr_3588] `EcucValueCollection.ecuExtract` always required

Imposition time: CP: `IT_EcucVal`

⌈The attribute `EcucValueCollection.ecuExtract` shall always be defined.⌋

[constr_3589] `EcucModuleConfigurationValues.ecucDefEdition` always required

Imposition time: CP: `IT_EcucVal`

⌈The attribute `EcucModuleConfigurationValues.ecucDefEdition` shall always be defined.⌋

[constr_3591] `EcucModuleConfigurationValues.definition` always required

Imposition time: CP: `IT_EcucVal`

⌈The attribute `EcucModuleConfigurationValues.definition` shall always be defined.⌋

[constr_4203] `EcucModuleConfigurationValues.container` ⌈`EcucModuleConfigurationValues.container` shall exist at least once⌋

An `EcucModuleConfigurationValues` without any `EcucContainerValue` would not have any effect on the Ecu configuration.

A valid `EcucValueCollection` needs to reference the System description (an `ECU Extract` referenced via `ecuExtract`) [1] that specifies the environment in which the configured ECU operates. Additionally, it references all Software Module configurations (see section 2.4.2) that are part of this ECU Configuration. It shall be noted that several `EcucValueCollection`s are allowed in the context of one `ECU Extract`.

Class	**EcucValueCollection**			
Note	This represents the anchor point of the ECU configuration description. **Tags:** atp.recommendedPackage=EcucValueCollections This Class is only used by the AUTOSAR Classic Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	EcucValueCollection			
ecucValue	EcucModule ConfigurationValues	*	ref	References to the configuration of individual software modules that are present on this ECU. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=ecucValue, ecucValue.ecucModule ConfigurationValues, ecucValue.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
ecuExtract	System	0..1	ref	Represents the extract of the System Configuration that is relevant for the ECU configured with that ECU Configuration Description. **Tags:** atp.Splitkey=ecuExtract

Table 2.39: EcucValueCollection

[TPS_ECUC_02141] Variable reference `EcucValueCollection.ecucValue` ⌈ The reference `EcucValueCollection.ecucValue` is subject to variant handling. The existence can be evaluated using the variant handling mechanism.⌋

2.4.2 Module Configurations

[TPS_ECUC_03016] `EcucModuleConfigurationValues` properties ⌈ The `EcucModuleConfigurationValues` subsumes all configuration objects that belong to one managed Software Module, namely Application Software Components, BSW modules, RTE and generic ECU Configuration artifacts (e.g. memory maps).⌋

[TPS_ECUC_02089] The content of `EcucModuleConfigurationValues` is splitable among several XML-Files ⌈ The `EcucModuleConfigurationValues` aggregates the `EcucContainerValue` with the role `container` and the stereotype ≪`atpSplitable`≫ which allows the content of a `EcucModuleConfigurationValues` to be split among several XML-Files.⌋

[TPS_ECUC_02119] Variable existence of container on value side ⌈ The aggregated `container` is subject to variant handling. The existence can be evaluated using the variant handling mechanism.⌋

[TPS_ECUC_03017] `EcucModuleConfigurationValues` reference to `BswImplementation` ⌈ If the `EcucModuleConfigurationValues` holds the configuration values of a BSW module, a reference to the according `BswImplementation` shall be provided.⌋

The reference is established to the `BswImplementation` because this is the most detailed information available for the configuration.

[TPS_ECUC_03035] Assignment of `EcucModuleConfigurationValues` to an `EcucModuleDef` ⌈ The reference `definition` assigns the `EcucModuleConfigurationValues` to the according `EcucModuleDef` it is depending on.⌋

Note: Previous versions of the standard defined a sorting order for Container-, Parameter-, and Reference-Values. This was removed from the standard. Nevertheless, the assumption is that any tool generating ECU Configuration Values ARXML files will have a tool dependent stable sorting order.

The index is defined in the EcucIndexableValue class. EcucParameterValue and EcucAbstractReferenceValue inherit from EcucIndexableValue.

Class	***EcucIndexableValue*** (abstract)			
Note	Used to support the specification of ordering of parameter values. This Class is only used by the AUTOSAR Classic Platform.			
Base	*ARObject*			
Subclasses	*EcucAbstractReferenceValue*, EcucContainerValue, *EcucParameterValue*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
index	PositiveInteger	0..1	attr	Used to support the creation of collections where the order has a semantic relevance. **Tags:** xml.sequenceOffset=-5

Table 2.40: EcucIndexableValue

[TPS_ECUC_06072] Container-, Parameter-, and Reference-Values with requiresIndex set to true ⌈ Container-, Parameter-, and Reference-Values which have the requiresIndex set to true in their definition shall provide an index.⌋

[TPS_ECUC_03031] EcucModuleDef includes standardized and vendor-specific parameter definitions ⌈ The EcucModuleDef, to which the EcucModuleConfigurationValues is associated to, is specified by the implementor of the according Software Module. Therefore the EcucModuleDef includes standardized as well as vendor-specific parameter definitions.⌋

Class	**EcucModuleConfigurationValues**			
Note	Head of the configuration of one Module. A Module can be a BSW module as well as the RTE and ECU Infrastructure. As part of the BSW module description, the EcucModuleConfigurationValues element has two different roles: The recommendedConfiguration contains parameter values recommended by the BSW module vendor. The preconfiguredConfiguration contains values for those parameters which are fixed by the implementation and cannot be changed. These two EcucModuleConfigurationValues are used when the base EcucModuleConfigurationValues (as part of the base ECU configuration) is created to fill parameters with initial values. **Tags:** atp.recommendedPackage=EcucModuleConfigurationValuess This Class is only used by the AUTOSAR Classic Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	**EcucModuleConfigurationValues**			
container	EcucContainerValue	*	aggr	Aggregates all containers that belong to this module configuration. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=container, container.shortName, container.variationPoint.shortLabel vh.latestBindingTime=postBuild xml.sequenceOffset=10
definition	EcucModuleDef	0..1	ref	Reference to the definition of this EcucModule ConfigurationValues element. Typically, this is a vendor specific module configuration. **Tags:** atp.Splitkey=definition xml.sequenceOffset=-10
ecucDefEdition	RevisionLabelString	0..1	attr	This is the version info of the ModuleDef ECUC Parameter definition to which this values conform to / are based on. For the Definition of ModuleDef ECUC Parameters the AdminData shall be used to express the semantic changes. The compatibility rules between the definition and value revision labels is up to the module's vendor.
implementation ConfigVariant	EcucConfiguration VariantEnum	0..1	attr	Specifies the kind of deliverable this EcucModule ConfigurationValues element provides. If this element is not used in a particular role (e.g. preconfigured Configuration or recommendedConfiguration) then the value shall be one of VariantPreCompile, VariantLink Time, VariantPostBuild. **Tags:** atp.Status=obsolete
module Description	BswImplementation	0..1	ref	Referencing the BSW module description, which this EcucModuleConfigurationValues element is configuring. This is optional because the EcucModuleConfiguration Values element is also used to configure the ECU infrastructure (memory map) or Application SW-Cs. However in case the EcucModuleConfigurationValues are used to configure the module, the reference is mandatory in order to fetch module specific "common" published information. **Tags:** atp.Splitkey=moduleDescription
postBuildVariant Used	Boolean	0..1	attr	Indicates whether a module implementation has or plans to have (i.e., introduced at link or post-build time) new post-build variation points. TRUE means yes, FALSE means no. If the attribute is not defined, FALSE semantics shall be assumed.

Table 2.41: EcucModuleConfigurationValues

Figure 2.29 depicts the different associations between the `EcucModuleConfigurationValues` and the `Basic Software Module Description`. The `BswImplementation` may specify a vendor specific pre-configured configuration Values (`preconfiguredConfiguration`) that includes the configuration values already assigned by the implementor of the Software Module and a vendor specific recommended configuration Values (`recommendedConfiguration`) that can be used to initialize configuration editors.

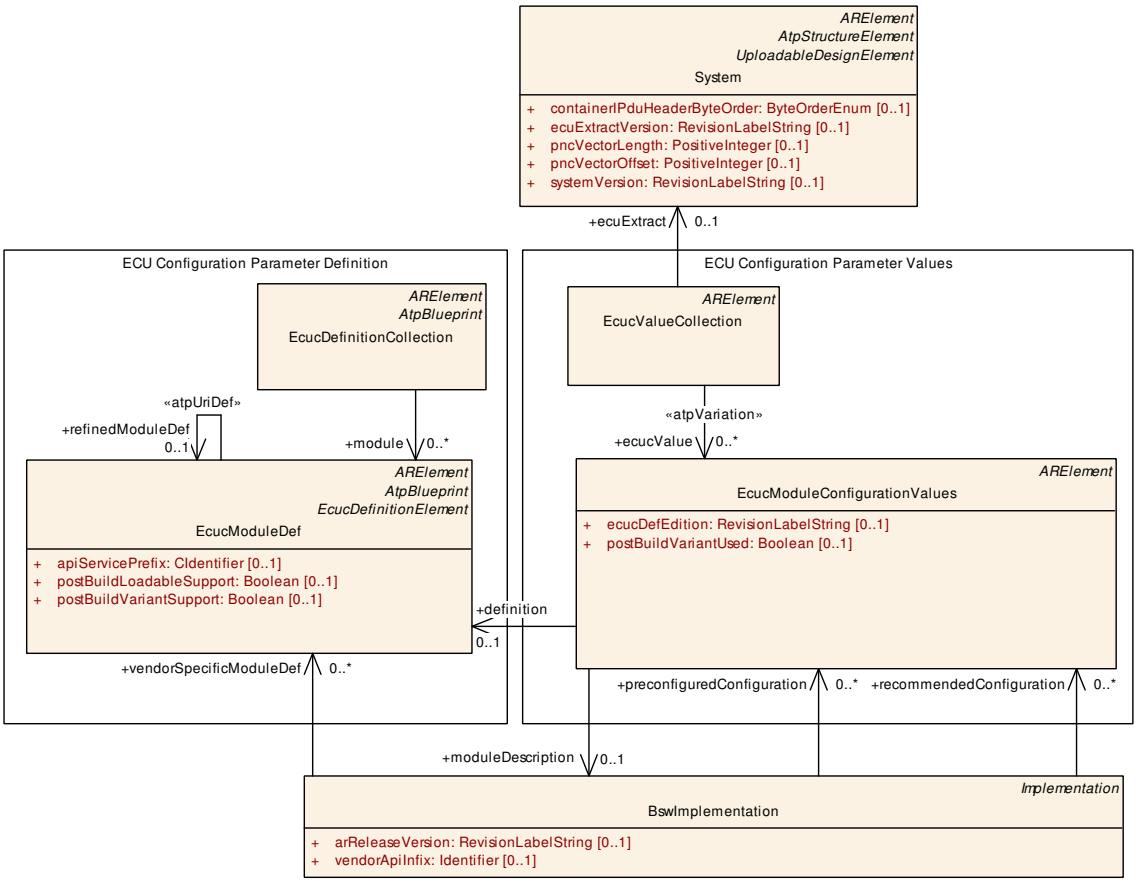

Figure 2.29: Dependencies of ModuleConfigurations

The element `supportedConfigVariant` is described in section 2.3.2 and section 2.3.4.3.2.

[TPS_ECUC_02147] Introducing new post build variants at post build configuration time ⌈In order to indicate that post build variants are intended to be added at post build configuration time in a specific module implementation, `postBuildVariantUsed` of the `EcucModuleConfigurationValues` shall be set to TRUE.⌋

[TPS_ECUC_01063] Impact of `postBuildVariantUsed` value set to FALSE ⌈If the value of the `EcucModuleConfigurationValues.postBuildVariantUsed` is set to FALSE or if it is not defined, it is not possible to add a post-build variant at post-build configuration time.⌋

[constr_3450] `postBuildVariantUsed` value in case of post build `VariationPoints`

Imposition time: CP: `IT_EcucVal`

⌈If the configuration values of a BSW module contain at least one post build `VariationPoint`, the value of the `postBuildVariantUsed` for the `EcucModuleConfigurationValues` shall be set to TRUE.⌋

[constr_3451] No `EcucModuleConfigurationValues.postBuildVariantUsed` value setting restriction in case `postBuildVariantSupport` is set to TRUE

Imposition time: CP: `IT_EcucVal`

⌈If `EcucModuleDef.postBuildVariantSupport` is set to TRUE, then `EcucModuleConfigurationValues.postBuildVariantUsed` can be either TRUE or FALSE.⌋

[constr_3452] `EcucModuleConfigurationValues.postBuildVariantUsed` value setting restriction in case `postBuildVariantSupport` is set to FALSE

Imposition time: CP: `IT_EcucVal`

⌈If `EcucModuleDef.postBuildVariantSupport` is set to FALSE, then `EcucModuleConfigurationValues.postBuildVariantUsed` shall be FALSE.⌋

To illustrate the structure of `ECU Configuration Values` Listing 2.26 depicts the top-level structure of an `ECU Configuration Values ARXML` file that conforms to the ECU Configuration Definition ARXML file that was presented in Listing 2.5. Please note that it is allowed to have an arbitrary number of packages before a module package definition (e.g. /AUTOSAR/Ecuc_VendorX/CanIf/...).

The only `supportedConfigVariant` of Listing 2.5 is taken for the `implementationConfigVariant` element.

```
<AR-PACKAGE>
  <SHORT-NAME>ECUC1</SHORT-NAME>
  <ELEMENTS>
    <ECUC-VALUE-COLLECTION>
      <SHORT-NAME>Configuration</SHORT-NAME>
      <ECU-EXTRACT-REF DEST="SYSTEM">/some_package/some_path/
          theEcuExtractForEcuXY</ECU-EXTRACT-REF>
      <ECUC-VALUES>
        <ECUC-MODULE-CONFIGURATION-VALUES-REF-CONDITIONAL>
          <ECUC-MODULE-CONFIGURATION-VALUES-REF DEST="ECUC-MODULE-
              CONFIGURATION-VALUES">/ECUC1/theRteConfig</ECUC-MODULE-
              CONFIGURATION-VALUES-REF>
        </ECUC-MODULE-CONFIGURATION-VALUES-REF-CONDITIONAL>
      </ECUC-VALUES>
    </ECUC-VALUE-COLLECTION>
    <ECUC-MODULE-CONFIGURATION-VALUES>
      <SHORT-NAME>theRteConfig</SHORT-NAME>
      <DEFINITION-REF DEST="ECUC-MODULE-DEF">/AUTOSAR/EcucDefs/Rte</
          DEFINITION-REF>
      <IMPLEMENTATION-CONFIG-VARIANT>VARIANT-PRE-COMPILE</IMPLEMENTATION-
          CONFIG-VARIANT>
      <MODULE-DESCRIPTION-REF DEST="BSW-IMPLEMENTATION">/some_package/
          some_path/theUsed_Rte_BSWModuleImplementation</MODULE-DESCRIPTION-
          REF>
      <CONTAINERS>
        <ECUC-CONTAINER-VALUE>
          <SHORT-NAME>theGeneration</SHORT-NAME>
          <DEFINITION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF">/AUTOSAR/
              EcucDefs/Rte/RteGeneration</DEFINITION-REF>
```

```
          <SUB-CONTAINERS>
            <!-- ... -->
          </SUB-CONTAINERS>
        </ECUC-CONTAINER-VALUE>
      </CONTAINERS>
    </ECUC-MODULE-CONFIGURATION-VALUES>
  </ELEMENTS>
</AR-PACKAGE>
```

Listing 2.26: `ECU Configuration Values ARXML` top-level structure

2.4.2.1 Splitable ModuleConfiguration

In the document *Generic Structure Template* [4] it is specified that the elements of an aggregation are allowed to be split over several XML files if the relationship is marked with the stereotype ≪atpSplitable≫.

The stereotype ≪atpSplitable≫ has been introduced to support the delivery of *one* module's EcucModuleConfigurationValues in *several* XML files, see also AUTOSAR Classic Platform Methodology [2] chapter 2.7.8.3 and 2.7.8.4 for use-cases.

Each splitable property (attribute, aggregation, reference) need to be uniquely identifiable. This happens usually by shortName. The DEFINITION-REF can also be used. For example, the EcucParameterValues of an EcucContainerValue are allowed to be split over several XML files. Each EcucParameterValue is uniquely identifiable via the reference to the EcucParameterDef. More details can be found in the Generic Structure Template [4].

In Listing 2.27 a simple definition of a module's configuration parameters is shown. It just consists of one container which has two parameters, one parameter defined to be PRE-COMPILE time configurable and the other parameter is POST-BUILD time configurable with respect to their value. The values for these parameters are defined in different process steps and therefore two XML files can be used to describe both values.

In Listing 2.28 the value for the PRE-COMPILE time parameter ComSignalLength is specified, while in Listing 2.29 the POST-BUILD parameter's ComSignalInitValue value is given.

The XML structure in both EcucModuleConfigurationValues XML files is equivalent with respect to the packages and containers. In both XML files a container with the name theSignal is defined. It is up to the configuration tool to *merge* the content of these two files into one model. Also is the number of possible XML files not limited, so it would be possible (although probably not reasonable) to put each parameter value into one XML file.

```
<ECUC-MODULE-DEF>
  <SHORT-NAME>Com</SHORT-NAME>
  <LOWER-MULTIPLICITY>0</LOWER-MULTIPLICITY>
  <UPPER-MULTIPLICITY>1</UPPER-MULTIPLICITY>
  <POST-BUILD-LOADABLE-SUPPORT>true</POST-BUILD-LOADABLE-SUPPORT>
  <POST-BUILD-VARIANT-SUPPORT>true</POST-BUILD-VARIANT-SUPPORT>
  <CONTAINERS>
    <ECUC-PARAM-CONF-CONTAINER-DEF>
      <SHORT-NAME>ComSignal</SHORT-NAME>
      <LOWER-MULTIPLICITY>0</LOWER-MULTIPLICITY>
      <UPPER-MULTIPLICITY>*</UPPER-MULTIPLICITY>
      <POST-BUILD-LOADABLE-MULTIPLICITY>true</POST-BUILD-LOADABLE-
          MULTIPLICITY>
      <POST-BUILD-VARIANT-MULTIPLICITY>true</POST-BUILD-VARIANT-
          MULTIPLICITY>
      <PARAMETERS>
        <ECUC-INTEGER-PARAM-DEF>
          <SHORT-NAME>ComSignalLength</SHORT-NAME>
          <ORIGIN>AUTOSAR_ECUC</ORIGIN>
          <POST-BUILD-LOADABLE-VALUE>false</POST-BUILD-LOADABLE-VALUE>
          <POST-BUILD-VARIANT-VALUE>false</POST-BUILD-VARIANT-VALUE>
        </ECUC-INTEGER-PARAM-DEF>
        <ECUC-INTEGER-PARAM-DEF>
          <SHORT-NAME>ComSignalInitValue</SHORT-NAME>
          <ORIGIN>AUTOSAR_ECUC</ORIGIN>
          <POST-BUILD-LOADABLE-VALUE>true</POST-BUILD-LOADABLE-VALUE>
          <POST-BUILD-VARIANT-VALUE>true</POST-BUILD-VARIANT-VALUE>
        </ECUC-INTEGER-PARAM-DEF>
      </PARAMETERS>
    </ECUC-PARAM-CONF-CONTAINER-DEF>
  </CONTAINERS>
</ECUC-MODULE-DEF>
```

Listing 2.27: Parameter Definition Example

```
<ECUC-MODULE-CONFIGURATION-VALUES>
  <SHORT-NAME>theComConfig</SHORT-NAME>
  <DEFINITION-REF DEST="ECUC-MODULE-DEF">/AUTOSAR/EcucDefs/Com</DEFINITION-
      REF>
  <MODULE-DESCRIPTION-REF DEST="BSW-IMPLEMENTATION">/some_package/some_path
      /theUsed_Com_BSWModuleImplementation</MODULE-DESCRIPTION-REF>
  <CONTAINERS>
    <ECUC-CONTAINER-VALUE>
      <SHORT-NAME>theSignal</SHORT-NAME>
      <DEFINITION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF">/AUTOSAR/
          EcucDefs/Com/ComSignal</DEFINITION-REF>
      <PARAMETER-VALUES>
        <ECUC-NUMERICAL-PARAM-VALUE>
          <DEFINITION-REF DEST="ECUC-INTEGER-PARAM-DEF">/AUTOSAR/EcucDefs/
              Com/ComSignal/ComSignalLength</DEFINITION-REF>
            <VALUE>2</VALUE>
          </ECUC-NUMERICAL-PARAM-VALUE>
      </PARAMETER-VALUES>
  </ECUC-CONTAINER-VALUE>
  </CONTAINERS>
```

```
</ECUC-MODULE-CONFIGURATION-VALUES>
```

Listing 2.28: Example of a split Ecu Configuration - part 1

```
<ECUC-MODULE-CONFIGURATION-VALUES>
  <SHORT-NAME>theComConfig</SHORT-NAME>
  <DEFINITION-REF DEST="ECUC-MODULE-DEF">/AUTOSAR/EcucDefs/Com</DEFINITION-
      REF>
  <MODULE-DESCRIPTION-REF DEST="BSW-IMPLEMENTATION">/some_package/some_path
      /theUsed_Com_BSWModuleImplementation</MODULE-DESCRIPTION-REF>
  <CONTAINERS>
    <ECUC-CONTAINER-VALUE>
      <SHORT-NAME>theSignal</SHORT-NAME>
      <DEFINITION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF">/AUTOSAR/
          EcucDefs/Com/ComSignal</DEFINITION-REF>
      <PARAMETER-VALUES>
        <ECUC-NUMERICAL-PARAM-VALUE>
          <DEFINITION-REF DEST="ECUC-INTEGER-PARAM-DEF">/AUTOSAR/EcucDefs/
              Com/ComSignal/ComSignalInitValue</DEFINITION-REF>
          <VALUE>0</VALUE>
        </ECUC-NUMERICAL-PARAM-VALUE>
      </PARAMETER-VALUES>
    </ECUC-CONTAINER-VALUE>
  </CONTAINERS>
</ECUC-MODULE-CONFIGURATION-VALUES>
```

Listing 2.29: Example of a split Ecu Configuration - part 2

2.4.3 Parameter Container Description

Symmetrically to the parameter container definition (see section 2.3.3) the parameter container description is specified to group other containers, parameter values and references. Figure 2.30 depicts the general structure of the configuration container values and its association to the configuration definition. The dependencies reflect the direct relationship between a `EcucContainerValue` and a `EcucContainerDef`, as well as a `EcucParameterValue` and a `EcucParameterDef` (parameter type).

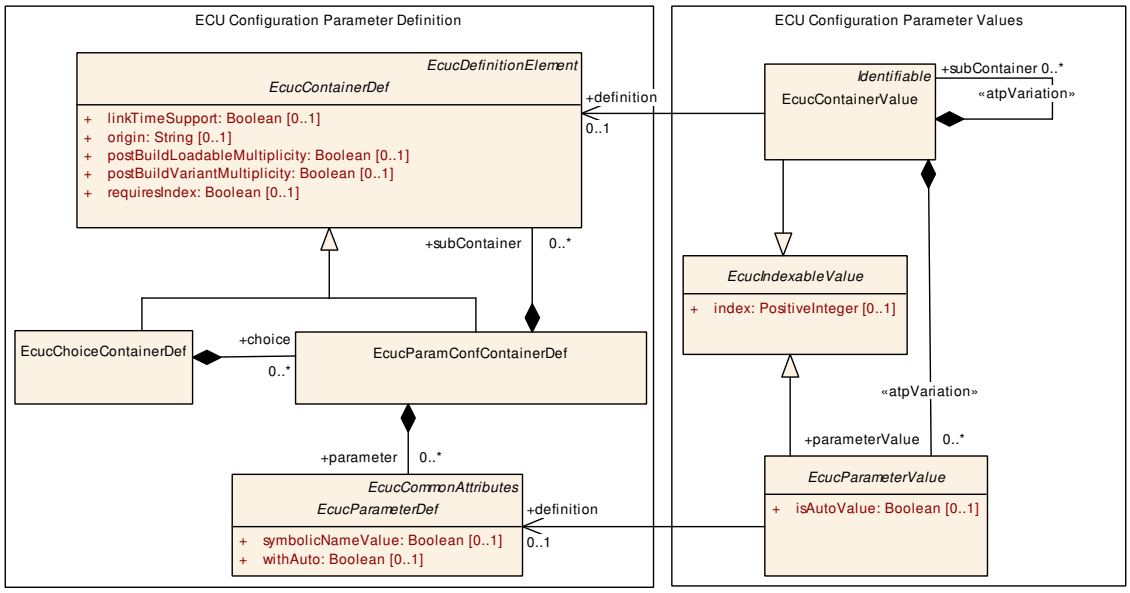

Figure 2.30: Parameter Container Values

[constr_3592] `EcucContainerValue.definition` always required

Imposition time: CP: `IT_EcucVal`

⌈The attribute `EcucContainerValue.definition` shall always be defined.⌋

[constr_3593] `EcucParameterValue.definition` always required

Imposition time: CP: `IT_EcucVal`

⌈The attribute `EcucParameterValue.definition` shall always be defined.⌋

[TPS_ECUC_03012] `EcucContainerValue` defines a namespace for all included containers, parameters and references ⌈ The `EcucContainerValue` inherits from `Identifiable` defining a namespace for all `EcucContainerValue`, `EcucParameterValue` and `EcucReferenceValue` that belong to that `EcucContainerValue`.⌋

[TPS_ECUC_08043] The number of `EcucContainerValue` instances in post-build loadable variants ⌈ECU configuration tools shall check that the number of `EcucContainerValue` instances of `EcucContainerDef`s with `postBuildLoadableMultiplicity` = `false` within identical `EcucContainerValue`s or `EcucModuleConfigurationValues`s (the qualified `shortName` path starting from the `shortName` of the `EcucModuleConfigurationValues` is the same) is the same in ECU configurations updated at post-build time.⌋

[TPS_ECUC_08044] The number of `EcucContainerValue` instances in post-build selectable variants ⌈ECU configuration tools shall check that the number of `EcucContainerValue` instances of `EcucContainerDef`s with `postBuildVariantMultiplicity` set to `false` within identical `EcucContainerValue`s or `EcucModuleConfigurationValues`s (the qualified `shortName` path starting from

the shortName of the EcucModuleConfigurationValues is the same) is the same in all variation points bound at post-build time.⌋

[TPS_ECUC_03019] EcucContainerValue definition reference ⌈ The reference definition assigns the EcucContainerValue to the according EcucContainerDef[15] it is depending on.⌋

If the ECU Configuration Values would be provided without an according ECU Configuration Parameter Definition, an editor could not reconstruct what kind of EcucContainerDef a EcucContainerValue is based upon.

[TPS_ECUC_03011] EcucContainerDefs with lowerMultiplicity $<$ 1 and the effect on the corresponding EcucContainerValues ⌈ If a EcucContainerDef has specified a lowerMultiplicity $<$ 1 the corresponding EcucContainerValue may be omitted in the ECU Configuration Values, because it is being treated as optional.⌋

[15] including all EcucContainerDef's descendants

Class	**EcucContainerValue**			
Note	Represents a Container definition in the ECU Configuration Description. This Class is only used by the AUTOSAR Classic Platform.			
Base	*ARObject*, *EcucIndexableValue*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	EcucContainerValue.subContainer, EcucModuleConfigurationValues.container			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
definition	EcucContainerDef	0..1	ref	Reference to the definition of this Container in the ECU Configuration Parameter Definition. **Tags:** atp.Splitkey=definition xml.sequenceOffset=-10
parameterValue	EcucParameterValue	*	aggr	Aggregates all ECU Configuration Values within this Container. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=parameterValue, parameterValue.variation Point.shortLabel vh.latestBindingTime=postBuild
referenceValue	EcucAbstractReference Value	*	aggr	Aggregates all References with this container. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=referenceValue, referenceValue.variation Point.shortLabel vh.latestBindingTime=postBuild
subContainer	EcucContainerValue	*	aggr	Aggregates all sub-containers within this container. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=subContainer, subContainer.shortName, sub Container.variationPoint.shortLabel vh.latestBindingTime=postBuild

Table 2.42: EcucContainerValue

[TPS_ECUC_02120] Variable `subContainers` ⌈ The aggregated `subContainer` is subject to variant handling. The existence can be evaluated using the variant handling mechanism.⌋

[TPS_ECUC_02121] Variable `parameterValues` ⌈ The aggregated `parameterValue` is subject to variant handling. The existence can be evaluated using the variant handling mechanism.⌋

[TPS_ECUC_02122] Variable `referenceValues` ⌈ The aggregated `referenceValue` is subject to variant handling. The existence can be evaluated using the variant handling mechanism.⌋

In Listing 2.30 a snippet of an `ECU Configuration Values ARXML` file is shown that conforms to the `ECU Configuration Parameter Definition` described in Listing 2.6. The container `RteGeneration` is specified to have an `upperMultiplicity` of `1`, so there can only be one `EcucContainerValue` representation. The container `SwComponentInstance` has an `upperMultiplicity` of `*`, so there can be several representations of this `EcucContainerValue`.

```
<ECUC-MODULE-CONFIGURATION-VALUES>
  <SHORT-NAME>theRteConfig</SHORT-NAME>
```

```
<DEFINITION-REF DEST="ECUC-MODULE-DEF">/AUTOSAR/EcucDefs/Rte</DEFINITION-
    REF>
<IMPLEMENTATION-CONFIG-VARIANT>VARIANT-PRE-COMPILE</IMPLEMENTATION-CONFIG
    -VARIANT>
<MODULE-DESCRIPTION-REF DEST="BSW-IMPLEMENTATION">/some_package/some_path
    /theUsed_Rte_BSWModuleImplementation</MODULE-DESCRIPTION-REF>
<CONTAINERS>
  <ECUC-CONTAINER-VALUE>
    <SHORT-NAME>theGeneration</SHORT-NAME>
    <DEFINITION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF">/AUTOSAR/
        EcucDefs/Rte/RteGeneration</DEFINITION-REF>
    <SUB-CONTAINERS>
      <!-- ... -->
    </SUB-CONTAINERS>
  </ECUC-CONTAINER-VALUE>
  <ECUC-CONTAINER-VALUE>
    <SHORT-NAME>SwcInstance1</SHORT-NAME>
    <DEFINITION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF">/AUTOSAR/
        EcucDefs/Rte/RteSwComponentInstance</DEFINITION-REF>
    <SUB-CONTAINERS>
      <!-- ... -->
    </SUB-CONTAINERS>
  </ECUC-CONTAINER-VALUE>
  <ECUC-CONTAINER-VALUE>
    <SHORT-NAME>SwcInstance2</SHORT-NAME>
    <DEFINITION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF">/AUTOSAR/
        EcucDefs/Rte/RteSwComponentInstance</DEFINITION-REF>
    <SUB-CONTAINERS>
      <!-- ... -->
    </SUB-CONTAINERS>
  </ECUC-CONTAINER-VALUE>
</CONTAINERS>
</ECUC-MODULE-CONFIGURATION-VALUES>
```

Listing 2.30: `EcucContainerValue` Example

2.4.3.1 Choice Containers

[TPS_ECUC_03020] **EcucChoiceContainerDef on the value side** ⌈ In the ECU Configuration Parameter Definition, the container choices are specified as part of the EcucChoiceContainerDef. On the Value side, a EcucChoiceContainerDef is treated as a usual container, though it depends on the upperMultiplicity of the EcucChoiceContainerDef how often the choice can be taken. Which choice has been taken is defined by the <DEFINITION-REF> of the <SUB-CONTAINER>.⌋

Listing 2.31 depicts the notation of a filled out EcucChoiceContainerDef as described in Listing 2.7.

For the myGwSource001 only one choice is possible, in this case the ComGwSignal has been selected.

For the second part (ComGwDestination) three choices have been taken, myGwDestination021 has chosen ComGwSignal, then myGwDestination022 has chosen ComGwDestinationDescription and then myGwDestination023 has chosen another ComGwSignal again.

```
<ECUC-MODULE-CONFIGURATION-VALUES>
  <SHORT-NAME>myChoiceExample</SHORT-NAME>
  <DEFINITION-REF DEST="ECUC-MODULE-DEF">/AUTOSAR/EcucDefs/Com</DEFINITION-
      REF>
  <CONTAINERS>
    <ECUC-CONTAINER-VALUE>
      <SHORT-NAME>ComGwMapping001</SHORT-NAME>
      <DEFINITION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF">/AUTOSAR/
          EcucDefs/Com/ComConfig/ComGwMapping</DEFINITION-REF>
      <SUB-CONTAINERS>
        <ECUC-CONTAINER-VALUE>
          <SHORT-NAME>myGwSource001</SHORT-NAME>
          <DEFINITION-REF DEST="ECUC-CHOICE-CONTAINER-DEF">/AUTOSAR/
              EcucDefs/Com/ComConfig/ComGwMapping/ComGwSource</DEFINITION-
              REF>
          <SUB-CONTAINERS>
            <ECUC-CONTAINER-VALUE>
              <SHORT-NAME>myGwSource001_1</SHORT-NAME>
              <DEFINITION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF">/AUTOSAR
                  /EcucDefs/Com/ComConfig/ComGwMapping/ComGwSource/
                  ComGwSignal</DEFINITION-REF>
              <!--...-->
            </ECUC-CONTAINER-VALUE>
          </SUB-CONTAINERS>
        </ECUC-CONTAINER-VALUE>
        <ECUC-CONTAINER-VALUE>
          <SHORT-NAME>myGwDestination021</SHORT-NAME>
          <DEFINITION-REF DEST="ECUC-CHOICE-CONTAINER-DEF">/AUTOSAR/
              EcucDefs/Com/ComConfig/ComGwMapping/ComGwDestination</
              DEFINITION-REF>
          <SUB-CONTAINERS>
            <ECUC-CONTAINER-VALUE>
              <SHORT-NAME>myGwDestination021a</SHORT-NAME>
```

```
            <DEFINITION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF">/AUTOSAR
                /EcucDefs/Com/ComConfig/ComGwMapping/ComGwDestination/
                ComGwSignal</DEFINITION-REF>
            <!--...-->
          </ECUC-CONTAINER-VALUE>
        </SUB-CONTAINERS>
      </ECUC-CONTAINER-VALUE>
      <ECUC-CONTAINER-VALUE>
        <SHORT-NAME>myGwDestination022</SHORT-NAME>
        <DEFINITION-REF DEST="ECUC-CHOICE-CONTAINER-DEF">/AUTOSAR/
            EcucDefs/Com/ComConfig/ComGwMapping/ComGwDestination</
            DEFINITION-REF>
        <SUB-CONTAINERS>
          <ECUC-CONTAINER-VALUE>
            <SHORT-NAME>myGwDestination022a</SHORT-NAME>
            <DEFINITION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF">/AUTOSAR
                /EcucDefs/Com/ComConfig/ComGwMapping/ComGwDestination/
                ComGwDestinationDescription</DEFINITION-REF>
            <!--...-->
          </ECUC-CONTAINER-VALUE>
        </SUB-CONTAINERS>
      </ECUC-CONTAINER-VALUE>
      <ECUC-CONTAINER-VALUE>
        <SHORT-NAME>myGwDestination023</SHORT-NAME>
        <DEFINITION-REF DEST="ECUC-CHOICE-CONTAINER-DEF">/AUTOSAR/
            EcucDefs/Com/ComConfig/ComGwMapping/ComGwDestination</
            DEFINITION-REF>
        <SUB-CONTAINERS>
          <ECUC-CONTAINER-VALUE>
            <SHORT-NAME>myGwDestination023a</SHORT-NAME>
            <DEFINITION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF">/AUTOSAR
                /EcucDefs/Com/ComConfig/ComGwMapping/ComGwDestination/
                ComGwSignal</DEFINITION-REF>
            <!--...-->
          </ECUC-CONTAINER-VALUE>
        </SUB-CONTAINERS>
      </ECUC-CONTAINER-VALUE>
    </SUB-CONTAINERS>
  </ECUC-CONTAINER-VALUE>
 </CONTAINERS>
</ECUC-MODULE-CONFIGURATION-VALUES>
```

Listing 2.31: `EcucChoiceContainerDef` Example

2.4.4 Parameter Values

In the ECU Configuration Parameter Definition individual elements for the different types of parameters exist (e.g. Boolean, Integer, String, see section 2.3.5). On the ECU Configuration Values side, this distinction is no longer needed, because every parameter value element references the corresponding definition element and therefore has its type bound.

However, there is a different distinction for the parameter values based on the variant handling implementation (see section 2.3.4.1), and the documentation support (see section 2.3.5.9).

[TPS_ECUC_03006] EcucParameterValue is the base class for all parameter values ⌈All meta-model classes specifying parameter values are derived from EcucParameterValue.⌋

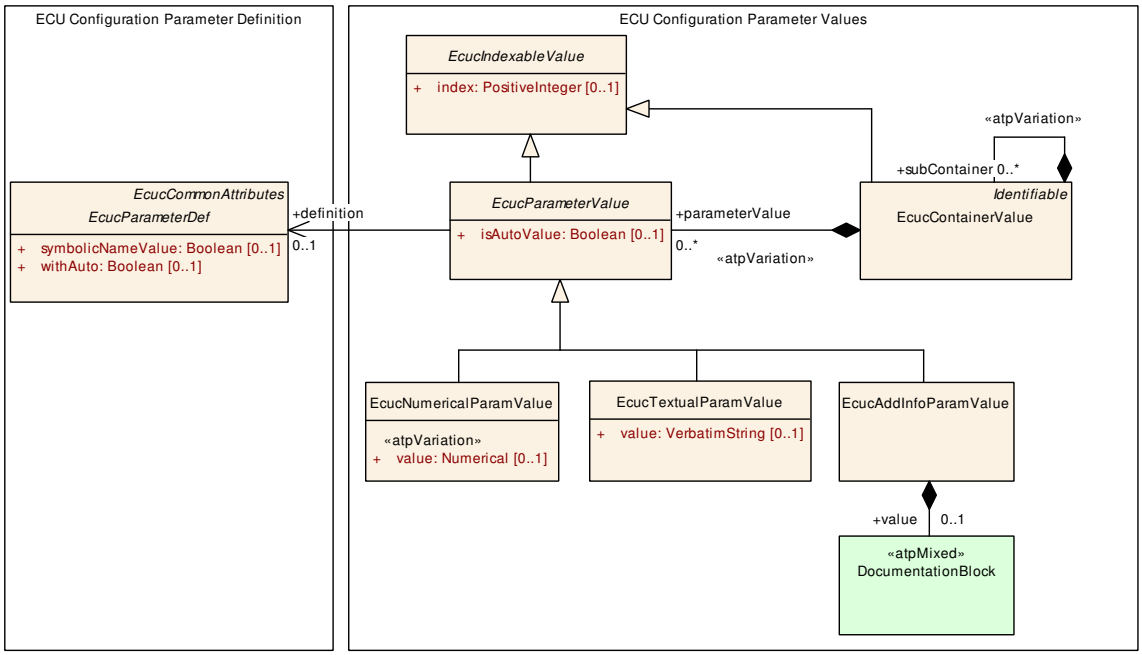

Figure 2.31: Parameter description

[constr_3594] EcucNumericalParamValue.value always required

Imposition time: CP: IT_EcucVal

⌈The attribute EcucNumericalParamValue.value shall always be defined.⌋

[constr_3595] EcucTextualParamValue.value always required

Imposition time: CP: IT_EcucVal

⌈The attribute EcucTextualParamValue.value shall always be defined.⌋

[constr_3596] `EcucAddInfoParamValue.value` always required

Imposition time: CP: `IT_EcucVal`

⌈The attribute `EcucAddInfoParamValue.value` shall always be defined.⌋

[TPS_ECUC_03007] Attribute `value` stores the configuration value in XML-based description ⌈All inheriting meta-model classes representing an ECU Configuration Value specify an attribute `value` that stores the configuration value in XML-based description.⌋

[TPS_ECUC_03038] Assignment of an `EcucParameterValue` to the corresponding `EcucParameterDef` ⌈The reference `definition` assigns the `EcucParameterValue`[16] to the according `EcucParameterDef` it is providing the value for.⌋

[TPS_ECUC_03009] A `defaultValue` that is specified in the `ECU Configuration Parameter Definition` may be used as the initial value in the `ECU Configuration Values` ⌈If a `defaultValue` is specified in the `ECU Configuration Parameter Definition` that given value can be used as the initial `value` of the according `EcucParameterValue` for the `ECU Configuration Values` as explained by [TPS_ECUC_01019].⌋

[TPS_ECUC_08054] Semantic of an optional parameter that is not present in the `ECU Configuration Values` ⌈The semantic of an optional parameter that is not present in the `ECU Configuration Values` is that there is no parameter value available, even if the `ECU Configuration Parameter Definition` provides a default value.⌋

[TPS_ECUC_03034] Each parameter in an `ECU Configuration Values` shall have a `value` ⌈In a well-formed and completed `ECU Configuration Values`, each provided parameter needs to have a `value` specified even if it is just copied from the `defaultValue` of the `ECU Configuration Parameter Definition`.⌋

For further rules how a `value` can be provided if no `defaultValue` is specified in the ECU Configuration Definition see section 4.2.

[TPS_ECUC_03010] Parameters that are declared as optional in the `ECU Configuration Parameter Definition` may be left out in the `ECU Configuration Values` ⌈If an ECU Configuration Parameter has specified a `lowerMultiplicity` < 1 an ECU Configuration Value may be left out in the `ECU Configuration Values` because of being treated as optional.⌋

[16] and all its sub-classes

Class	***EcucParameterValue*** (abstract)			
Note	Common class to all types of configuration values. This Class is only used by the AUTOSAR Classic Platform.			
Base	*ARObject*, *EcucIndexableValue*			
Subclasses	EcucAddInfoParamValue, EcucNumericalParamValue, EcucTextualParamValue			
Aggregated by	EcucContainerValue.parameterValue			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
annotation	Annotation	*	aggr	Possibility to provide additional notes while defining the ECU Configuration Parameter Values. These are not intended as documentation but are mere design notes. **Tags:** atp.Splitkey=annotation xml.sequenceOffset=10
definition	EcucParameterDef	0..1	ref	Reference to the definition of this EcucParameterValue subclasses in the ECU Configuration Parameter Definition. **Tags:** atp.Splitkey=definition xml.sequenceOffset=-10
isAutoValue	Boolean	0..1	attr	If withAuto is set to "true" for this parameter definition the isAutoValue can be set to "true". If isAutoValue is set to "true" the actual value will not be considered during ECU Configuration but will be (re-)calculated by the code generator and stored in the value attribute afterwards. These implicit updated values might require a re-generation of other modules which reference these values. If isAutoValue is not present the default is "false". **Tags:** xml.sequenceOffset=20

Table 2.43: EcucParameterValue

[TPS_ECUC_08045] The value of `EcucParameterValue` instances in post-build loadable variants ⌈ECU configuration tools shall check that the value of `EcucParameterValue` instances of `EcucParameterDef`s with `postBuildLoadableValue` set to `false` within identical `EcucContainerValue`s (the qualified `shortName` path starting from the `shortName` of the `EcucModuleConfigurationValues` is the same) is the same in ECU configurations updated at post-build time.⌋

[TPS_ECUC_08046] The value of `EcucParameterValue` instances in post-build selectable variants ⌈ECU configuration tools shall check that the value of `EcucParameterValue` instances of `EcucParameterDef`s with `postBuildVariantValue` set to `false` within identical `EcucContainerValue`s (the qualified `shortName` path starting from the `shortName` of the `EcucModuleConfigurationValues` is the same) is the same in all variation points bound at post-build time.⌋

[TPS_ECUC_08047] The number of `EcucParameterValue` instances in post-build loadable variants ⌈ECU configuration tools shall check that the number of `EcucParameterValue` instances of `EcucParameterDef`s with `postBuildLoadableMultiplicity` set to `false` within identical `EcucContainerValue`s (the qualified `shortName` path starting from the `shortName` of the `EcucModuleConfigurationValues` is the same) is the same in ECU configurations updated at post-build time.⌋

[TPS_ECUC_08048] The number of EcucParameterValue instances in post-build selectable variants ⌈ECU configuration tools shall check that the number of EcucParameterValue instances of EcucParameterDefs with postBuildVariantMultiplicity set to false within identical EcucParameterValues (the qualified shortName path starting from the shortName of the EcucModuleConfigurationValues is the same) is the same in all variation points bound at post-build time.⌋

[TPS_ECUC_08002] Introduction of new EcucParamConfContainerDef instances in post-build variants ⌈If a new EcucParamConfContainerDef instance is introduced in an updated post-build configuration, each EcucParameterValue and EcucReferenceValue within that EcucParamConfContainerDef instance and its aggregated EcucParamConfContainerDefs instanced in the role subContainer may be assigned a new value and have arbitrary number of instances (if upperMultiplicity is greater than lowerMultiplicity) regardless if its postBuildLoadableMultiplicity and postBuildLoadableValue settings.⌋

Example: HandleId value of an existing ComIPdu shall not be changed at post-build time as it is precompile configurable. However if a new ComIPdu instance is introduced at post-build time, it shall receive a new HandleId value. This basically means that postBuildLoadableMultiplicity and postBuildLoadableValue attributes are applicable only to parameters and references in container instances which already exist in the initial configuration before the post-build updates.

[TPS_ECUC_01064] Introduction of new EcucParameterValues of type EcucFunctionNameDef at post-build time ⌈In case a new EcucParameterValue of type EcucFunctionNameDef (see [TPS_ECUC_02033]) is introduced at post-build time, its value shall be one of the existing function names (e.g. callouts) defined in the Basic Software Module Description. This means that it is not allowed to introduce new functions at post-build time.⌋

2.4.4.1 Textual Parameter Value

For the storage of values of parameters which do not have a numerical representation the element EcucTextualParamValue shall be used.

[TPS_ECUC_02126] Values for parameter types stored in the element EcucTextualParamValue ⌈ Values for parameter types

- EcucEnumerationParamDef
- EcucAbstractStringParamDef and its sub-classes

shall be stored in the element EcucTextualParamValue.⌋

The actual value is stored in the element value as VerbatimString and shall conform to the definition of the ECU Configuration Parameter Definition which

is referenced in the definition element. The restrictions on the textual representation specified in section 2.3.5.4, section 2.3.5.5, section 2.3.5.6 and section 2.3.5.7 are applicable to the corresponding value specifications.

In case the value of the EcucTextualParamValue shall be affected by the variant handling, the existence of the individual alternative EcucTextualParamValue elements shall be made variant. The value element itself can not be affected by variant handling.

Class	**EcucTextualParamValue**			
Note	Holding a value which is not subject to variation. This Class is only used by the AUTOSAR Classic Platform.			
Base	*ARObject*, *EcucIndexableValue*, *EcucParameterValue*			
Aggregated by	EcucContainerValue.parameterValue			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
value	VerbatimString	0..1	attr	Value of the parameter, not subject to variant handling.

Table 2.44: EcucTextualParamValue

2.4.4.1.1 Examples of EcucTextualParamValue

Listing 2.32 depicts the ECU Configuration Values ARXML of definition type EcucLinkerSymbolDef for Listing 2.12.

```
<ECUC-TEXTUAL-PARAM-VALUE>
  <DEFINITION-REF DEST="ECUC-LINKER-SYMBOL-DEF">/AUTOSAR/EcucDefs/Rte/
    Resource/Pim/RtePimInitializationSymbol</DEFINITION-REF>
  <VALUE>MyPimInitValuesLightMaster</VALUE>
</ECUC-TEXTUAL-PARAM-VALUE>
```

Listing 2.32: Linker Symbol Value Example

Listing 2.33 depicts the ECU Configuration Values ARXML of definition type EcucFunctionNameDef for Listing 2.13.

```
<ECUC-TEXTUAL-PARAM-VALUE>
  <DEFINITION-REF DEST="ECUC-FUNCTION-NAME-DEF">/AUTOSAR/EcucDefs/Eep/
    EepInitConfiguration/EepJobEndNotification</DEFINITION-REF>
  <VALUE>Eep_VendorXY_JobEndNotification</VALUE>
</ECUC-TEXTUAL-PARAM-VALUE>
```

Listing 2.33: Function Name Value Example

Listing 2.34 depicts the ECU Configuration Values ARXML of definition type EcucEnumerationParamDef for Listing 2.14.

```
<ECUC-TEXTUAL-PARAM-VALUE>
  <DEFINITION-REF DEST="ECUC-ENUMERATION-PARAM-DEF">/AUTOSAR/EcucDefs/Rte/
    RteGeneration/RteGenerationMode</DEFINITION-REF>
  <VALUE>CompatibilityMode</VALUE>
</ECUC-TEXTUAL-PARAM-VALUE>
```

Listing 2.34: Enumeration Value Example

2.4.4.2 Numerical Parameter Value

If the value of a configuration parameter shall be provided as subject to variant handling the element EcucNumericalParamValue shall be used. The value element of EcucNumericalParamValue is defined as <<atpVariation>> (see section 2.3.4.1).

Class	**EcucNumericalParamValue**			
Note	Holding the value which is subject to variant handling. This Class is only used by the AUTOSAR Classic Platform.			
Base	*ARObject*, *EcucIndexableValue*, *EcucParameterValue*			
Aggregated by	EcucContainerValue.parameterValue			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
value	Numerical	0..1	attr	Value which is subject to variant handling. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime

Table 2.45: EcucNumericalParamValue

[TPS_ECUC_02142] Variable value of EcucNumericalParamValue.value ⌈ The value of EcucNumericalParamValue.value is subject to variant handling.⌋

2.4.4.2.1 Examples of EcucNumericalParamValue

Listing 2.35 depicts the ECU Configuration Values ARXML of definition type EcucBooleanParamDef for Listing 2.9.

```
<ECUC-NUMERICAL-PARAM-VALUE>
  <DEFINITION-REF DEST="ECUC-BOOLEAN-PARAM-DEF">/AUTOSAR/EcucDefs/Rte/
      RteGeneration/RteDevErrorDetect</DEFINITION-REF>
  <VALUE>1</VALUE>
</ECUC-NUMERICAL-PARAM-VALUE>
```

Listing 2.35: Boolean Value Example

Listing 2.36 depicts the ECU Configuration Values ARXML of definition type EcucIntegerParamDef for Listing 2.10.

```
<ECUC-TEXTUAL-PARAM-VALUE>
  <DEFINITION-REF DEST="ECUC-FUNCTION-NAME-DEF">/AUTOSAR/EcucDefs/Eep/
      EepInitConfiguration/EepJobEndNotification</DEFINITION-REF>
  <VALUE>Eep_VendorXY_JobEndNotification</VALUE>
</ECUC-TEXTUAL-PARAM-VALUE>
```

Listing 2.36: Integer Value Example

Listing 2.37 depicts the ECU Configuration Values ARXML of definition type EcucFloatParamDef for Listing 2.11.

```
<ECUC-NUMERICAL-PARAM-VALUE>
  <DEFINITION-REF DEST="ECUC-FLOAT-PARAM-DEF">/AUTOSAR/EcucDefs/Rte/
      RunnableEntityMapping/SchedulingPeriod</DEFINITION-REF>
```

```
  <VALUE>74.8</VALUE>
</ECUC-NUMERICAL-PARAM-VALUE>
```

Listing 2.37: Float Value Example

2.4.4.3 AddInfo Parameter Value

The only type-specific distinction for the values is the ECU Configuration Parameter Type EcucAddInfoParamDef (see section 2.3.5.9).

[TPS_ECUC_02123] The value of the parameter type EcucAddInfoParamDef ⌈ The value of the parameter type EcucAddInfoParamDef shall be provided in the element EcucAddInfoParamValue. This allows the usage of formatted text (see AUTOSAR Generic Structure Template [4] for further information).⌋

Class	**EcucAddInfoParamValue**			
Note	This parameter corresponds to EcucAddInfoParamDef. This Class is only used by the AUTOSAR Classic Platform.			
Base	*ARObject*, *EcucIndexableValue*, *EcucParameterValue*			
Aggregated by	EcucContainerValue.parameterValue			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
value	DocumentationBlock	0..1	aggr	Holds the content of the formated text. **Tags:** atp.Splitkey=value

Table 2.46: EcucAddInfoParamValue

Listing 2.38 depicts the ECU Configuration Values ARXML of definition type EcucAddInfoParamDef for Listing 2.15.

```
<ECUC-ADD-INFO-PARAM-VALUE>
  <DEFINITION-REF DEST="ECUC-ADD-INFO-PARAM-DEF">/AUTOSAR/EcucDefs/Dcm/
     DcmConfigSet/Dtc</DEFINITION-REF>
  <VALUE>
    <P>
      <L-1 L="EN">Description of the Dtc 0815.</L-1>
    </P>
  </VALUE>
</ECUC-ADD-INFO-PARAM-VALUE>
```

Listing 2.38: EcucAddInfoParamDef Example

2.4.5 References in the ECU Configuration Meta-model

Figure 2.32 depicts the ECU Configuration Meta-model to reference other description elements.

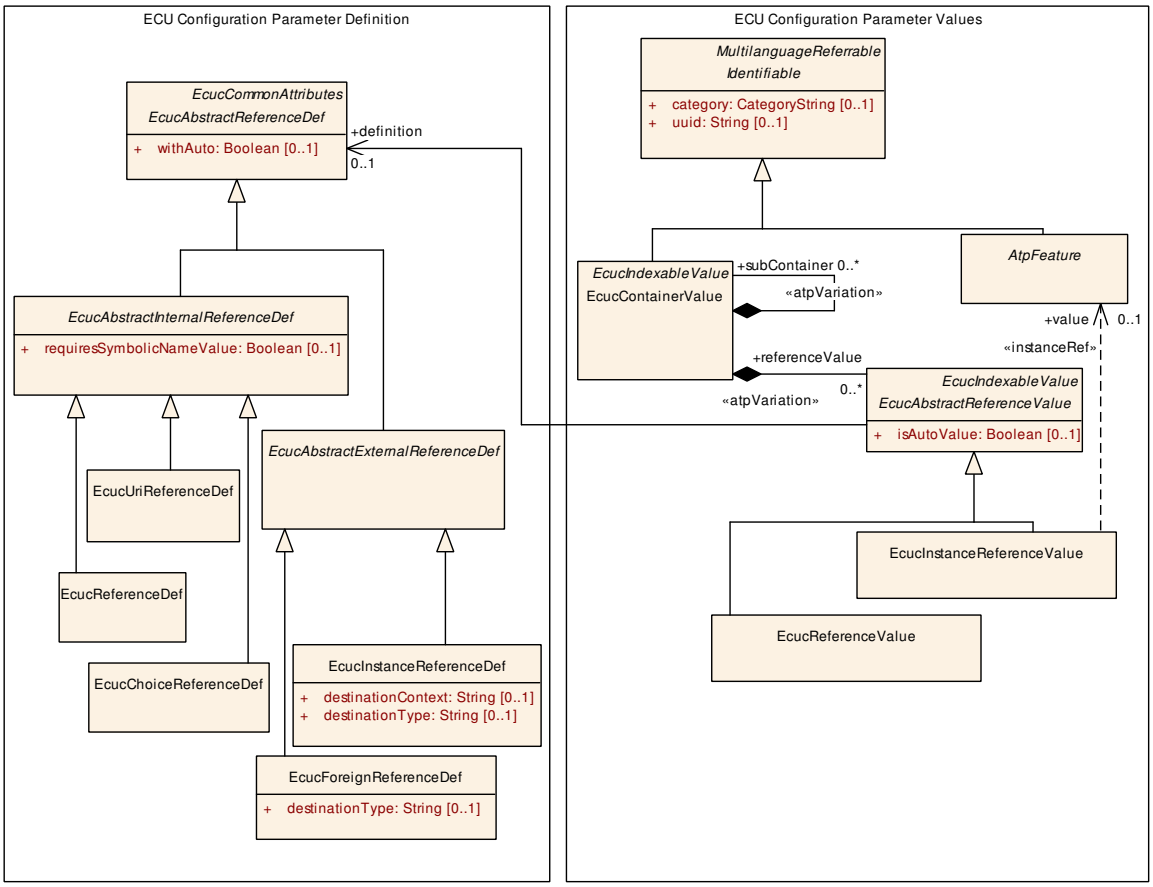

Figure 2.32: Parameter references

[constr_3597] `EcucAbstractReferenceValue.definition` always required

Imposition time: CP: `IT_EcucVal`

⌈The attribute `EcucAbstractReferenceValue.definition` shall always be defined.⌋

[constr_3598] `EcucInstanceReferenceValue.value` always required

Imposition time: CP: `IT_EcucVal`

⌈The attribute `EcucInstanceReferenceValue.value` shall always be defined.⌋

[constr_3599] `EcucReferenceValue.value` always required

Imposition time: CP: `IT_EcucVal`

⌈The attribute `EcucReferenceValue.value` shall always be defined.⌋

[TPS_ECUC_03032] Generalization of all reference types ⌈ The meta-model class `EcucAbstractReferenceValue` acts as the generalization of all reference types in the `ECU Configuration Values`.⌋

[TPS_ECUC_03039] `EcucAbstractReferenceValue definition` reference ⌈ The reference `definition` assigns the `EcucAbstractReferenceValue`[17] to the according `EcucAbstractReferenceDef` it is depending on.⌋

[TPS_ECUC_03030] `EcucAbstractReferenceDef`s with `lowerMultiplicity` < 1 and the effect on the corresponding `EcucAbstractReferenceValue`s ⌈ If a `EcucAbstractReferenceDef` has specified a `lowerMultiplicity` < 1 an according `EcucAbstractReferenceValue` may be omitted in the `ECU Configuration Values` because of being treated as optional.⌋

Class	***EcucAbstractReferenceValue*** (abstract)			
Note	Abstract class to be used as common parent for all reference values in the ECU Configuration Description. This Class is only used by the AUTOSAR Classic Platform.			
Base	*ARObject*, *EcucIndexableValue*			
Subclasses	EcucInstanceReferenceValue, EcucReferenceValue			
Aggregated by	EcucContainerValue.referenceValue			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
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:** atp.Splitkey=annotation
definition	EcucAbstractReference Def	0..1	ref	Reference to the definition of this EcucAbstractReference Value subclasses in the ECU Configuration Parameter Definition. **Tags:** atp.Splitkey=definition xml.sequenceOffset=-10
isAutoValue	Boolean	0..1	attr	If withAuto is set to "true" for this parameter definition the isAutoValue can be set to "true". If isAutoValue is set to "true" the actual value will not be considered during ECU Configuration but will be (re-)calculated by the code generator and stored in the value attribute afterwards. These implicit updated values might require a re-generation of other modules which reference these values. If isAutoValue is not present the default is "false".

Table 2.47: EcucAbstractReferenceValue

[TPS_ECUC_03027] `EcucReferenceValue` provides the mechanism to reference model elements that are `Referrable` ⌈ The meta-model class `EcucReferenceValue` provides the mechanism to reference to any model element of type `Referrable`.⌋

[TPS_ECUC_03028] `EcucReferenceValue` describes `EcucReferenceDef`s, `EcucChoiceReferenceDef`s, and `EcucForeignReferenceDef`s in the `ECU Configuration Values` ⌈ `EcucReferenceValue` provides the means to describe all kinds of reference definitions except an `EcucInstanceReferenceDef`, which is described by [TPS_ECUC_03033] in more detail.⌋

[17] and all its descendants

[TPS_ECUC_03029] **EcucChoiceReferenceDef translates to a EcucReferenceValue in the ECU Configuration Values** ⌈ A EcucChoiceReferenceDef translates to a EcucReferenceValue in the ECU Configuration Values because the choice has to be resolved in that description. Therefore no special configuration Values type is introduced.⌋

Class	**EcucReferenceValue**			
Note	Used to represent a configuration value that has a parameter definition of type EcucAbstractReference Def (used for all of its specializations excluding EcucInstanceReferenceDef). This Class is only used by the AUTOSAR Classic Platform.			
Base	*ARObject*, *EcucAbstractReferenceValue*, *EcucIndexableValue*			
Aggregated by	EcucContainerValue.referenceValue			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
value	Referrable	0..1	ref	Specifies the destination of the reference. **Tags:** atp.Splitkey=value

Table 2.48: EcucReferenceValue

[TPS_ECUC_08049] The value of EcucAbstractReferenceValue instances in post-build loadable variants ⌈ECU configuration tools shall check that the value of EcucAbstractReferenceValue instances of EcucAbstractReferenceDefs with postBuildLoadableValue set to false within identical EcucContainerValues (the qualified shortName path starting from the shortName of the EcucModuleConfigurationValues is the same) is the same in all variation points bound at post-build time.⌋

[TPS_ECUC_08050] The value of EcucAbstractReferenceValue instances in post-build selectable variants ⌈ECU configuration tools shall check that the value of EcucAbstractReferenceValue instances of EcucAbstractReferenceDefs with postBuildVariantValue set to false within identical EcucContainerValues (the qualified shortName path starting from the shortName of the EcucModuleConfigurationValues is the same) is the same in all variation points bound at post-build time.⌋

[TPS_ECUC_08051] The number of EcucAbstractReferenceValue instances in post-build loadable variants ⌈ECU configuration tools shall check that the number of EcucAbstractReferenceValue instances of EcucAbstractReferenceDefs with postBuildLoadableMultiplicity set to false within identical EcucParameterValues (the qualified shortName path starting from the shortName of the EcucModuleConfigurationValues is the same) is the same in all variation points bound at post-build time.⌋

[TPS_ECUC_08052] The number of EcucAbstractReferenceValue instances in post-build selectable variants ⌈ECU configuration tools shall check that the number of EcucAbstractReferenceValue instances of EcucAbstractReferenceDefs with postBuildVariantMultiplicity set to false within identical EcucParameterValues (the qualified shortName path starting from the shortName of

the EcucModuleConfigurationValues is the same) is the same in all variation points bound at post-build time.⌋

[TPS_ECUC_02093] Referenced containers shall be part of the same EcucValueCollection as the reference itself ⌈ If a EcucAbstractReferenceValue references a container within some EcucModuleConfigurationValues the referenced container shall be part of a EcucModuleConfigurationValues which is itself part of the EcucValueCollection.⌋

According to figure 2.28 a EcucModuleConfigurationValues is part of the EcucValueCollection if it is referenced with the ecucValue role.

The following examples will picture that EcucReferenceValue can be used to represent most of the specializations of EcucAbstractReferenceDef (namely EcucReferenceDef, EcucChoiceReferenceDef, and EcucForeignReferenceDef).

Listing 2.39 depicts the ECU Configuration Values ARXML of definition type EcucReferenceDef for Listing 2.16.

```
<ECUC-REFERENCE-VALUE>
  <DEFINITION-REF DEST="ECUC-REFERENCE-DEF">/AUTOSAR/EcucDefs/Os/
     OsApplication/OsAppScheduleTableRef</DEFINITION-REF>
  <VALUE-REF DEST="ECUC-CONTAINER-VALUE">/ECUC/myOs/myOsScheduleTable1</
     VALUE-REF>
</ECUC-REFERENCE-VALUE>
```

Listing 2.39: EcuC Reference Value Example (reference to an EcucContainerValue)

Listing 2.40 depicts the ECU Configuration Values ARXML of definition type EcucChoiceReferenceDef for Listing 2.17. To illustrate the usage of a EcucChoiceReferenceDef in more detail, this example takes advantage of the fact that a PortPin may be used in several modes at once. Therefore it has multiple references of different type.

```
<ECUC-REFERENCE-VALUE>
  <DEFINITION-REF DEST="ECUC-CHOICE-REFERENCE-DEF">/AUTOSAR/EcucDefs/Port/
     PortPin/PortPinMode</DEFINITION-REF>
  <VALUE-REF DEST="ECUC-CONTAINER-VALUE">/ECUC/mySpi/aSpiExternalDevice1</
     VALUE-REF>
</ECUC-REFERENCE-VALUE>
```

Listing 2.40: Choice Reference Example (reference to one of several possible target types)

Listing 2.41 depicts the ECU Configuration Values ARXML of definition type EcucForeignReferenceDef for Listing 2.18.

```
<ECUC-REFERENCE-VALUE>
  <DEFINITION-REF DEST="ECUC-FOREIGN-REFERENCE-DEF">/AUTOSAR/EcucDefs/Rte/
     Communication/FrameMapping/SystemFrame</DEFINITION-REF>
  <VALUE-REF DEST="FRAME">/SystemDescription/SystemFrameNo42</VALUE-REF>
</ECUC-REFERENCE-VALUE>
```

Listing 2.41: Foreign Reference Value Example (reference to any other model element)

2.4.5.1 Instance Reference Values

Due to the formalization of prototypes in the AUTOSAR Templates (see [4]) the reference to the instance of a prototype needs to declare the complete context in which the instance is residing.

[TPS_ECUC_03033] `EcucInstanceReferenceValue` provides the mechanism to reference an instance of a prototype ⌈ The meta-model class `EcucInstanceReferenceValue` provides the mechanism to reference to an actual instance of a prototype. This is achieved by specifying a relation with the stereotype <<*instanceRef*>>.⌋

In figure 2.33 the detailed modeling of the `EcucInstanceReferenceValue` <<*instanceRef*>> is specified.

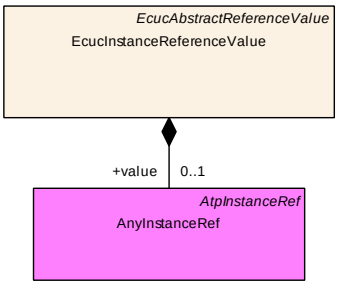

Figure 2.33: Instance Reference Value details

Class	**EcucInstanceReferenceValue**			
Note	InstanceReference representation in the ECU Configuration. This Class is only used by the AUTOSAR Classic Platform.			
Base	*ARObject*, *EcucAbstractReferenceValue*, *EcucIndexableValue*			
Aggregated by	EcucContainerValue.referenceValue			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
value	AtpFeature	0..1	iref	InstanceReference representation in the ECU Configuration. **Tags:** atp.Splitkey=value, value.contextElement, value.target **InstanceRef implemented by:** AnyInstanceRef

Table 2.49: EcucInstanceReferenceValue

Listing 2.42 depicts the `ECU Configuration Values ARXML` of definition type `EcucInstanceReferenceDef` for Listing 2.19. As one can see in the example the reference value is decomposed of the context path of the instance and the reference to the instance itself.

```
<ECUC-INSTANCE-REFERENCE-VALUE>
  <DEFINITION-REF DEST="ECUC-INSTANCE-REFERENCE-DEF">/AUTOSAR/EcucDefs/Rte/
     DataMappings/DataSRMapping/DataElementPrototypeRef</DEFINITION-REF>
  <VALUE-IREF>
    <CONTEXT-ELEMENT-REF DEST="SW-COMPONENT-PROTOTYPE">/SoftwareComponents/
       RootComposition/DoorFr</CONTEXT-ELEMENT-REF>
    <CONTEXT-ELEMENT-REF DEST="R-PORT-PROTOTYPE">/SoftwareComponents/
       DoorType/DoorAntennaReceiver</CONTEXT-ELEMENT-REF>
```

```
    <TARGET-REF DEST="VARIABLE-DATA-PROTOTYPE">/PortInterfaces/
       DoorAntennaInterface/AntennaStatus</TARGET-REF>
  </VALUE-IREF>
</ECUC-INSTANCE-REFERENCE-VALUE>
```

Listing 2.42: Instance Reference Example

The usage of ImplementationDataTypes within an AnyInstanceRef is described in detail in [4].

2.4.5.2 Representation of Symbolic Names

[constr_3217] Symbolic name reference shall point only to containers with a symbolic name value defined

Imposition time: CP: IT_EcucVal

⌈If an EcucReferenceValue exists that refers in the role definition to an EcucAbstractInternalReferenceDef with the attribute requiresSymbolicNameValue set to true, then the EcucContainerValue that is the target of the reference shall refer to an EcucParamConfContainerDef in the role definition that contains a definition of an EcucParameterDef where the attribute symbolicNameValue exists and is set to true. The EcucContainerValue shall define an EcucParameterValue that refers to an EcucParameterDef where the attribute symbolicNameValue exists and is set to true.⌋

Note: In other words if a symbolic name reference points to a container this container shall have a symbolic name value defined.

Please note that [constr_3217] also applies to EcucReferenceValues of EcucUriReferenceDefs although the target of the reference is determined by matching destinationUri.

[TPS_ECUC_03037] Deriving the symbolic name in the implementation from the shortName of the referenced container ⌈ The symbolic name in the implementation is expected to be derived later on from the shortName of the referenced destination according to [TPS_ECUC_02108].⌋

[TPS_ECUC_02107] Values of parameters with the symbolicNameValue set to TRUE that are assigned by the configuration editor or module generator shall be stored in the XML file ⌈ Configuration parameter values which represent symbolic name values shall be stored in the corresponding XML file after the configuration editor or module generator assigned the actual value.⌋

Listing 2.43 depicts the ECU Configuration Values ARXML of definition type EcucReferenceDef with requiresSymbolicNameValue set to true for Listing 2.20. To give a better impression how the referencing mechanism and code generation may work the EcucModuleConfigurationValues of the using and the providing modules are shown here.

```
<ECUC-MODULE-CONFIGURATION-VALUES>
  <SHORT-NAME>myCorTst</SHORT-NAME>
  <DEFINITION-REF DEST="ECUC-MODULE-DEF">/AUTOSAR/EcucDefs/CorTst</
      DEFINITION-REF>
  <CONTAINERS>
    <ECUC-CONTAINER-VALUE>
      <SHORT-NAME>Dem_PLL_lock_error</SHORT-NAME>
      <DEFINITION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF">/AUTOSAR/
          EcucDefs/CorTst/CorTstDemEventParameterRefs</DEFINITION-REF>
      <REFERENCE-VALUES>
        <ECUC-REFERENCE-VALUE>
          <DEFINITION-REF DEST="ECUC-REFERENCE-DEF">/AUTOSAR/EcucDefs/
              CorTst/CorTstDemEventParameterRefs/CORTST_E_CORE_FAILURE</
              DEFINITION-REF>
          <VALUE-REF DEST="ECUC-CONTAINER-VALUE">/ECUC/myDem/
              CORTST_E_CORE_FAILURE_1</VALUE-REF>
        </ECUC-REFERENCE-VALUE>
      </REFERENCE-VALUES>
    </ECUC-CONTAINER-VALUE>
  </CONTAINERS>
</ECUC-MODULE-CONFIGURATION-VALUES>
```

Listing 2.43: Symbolic Name Reference Example

[TPS_ECUC_02108] Rule for the creation of `#define` symbols in the header file for parameters with the `symbolicNameValue` set to `TRUE` ⌈The values of EcucParameterDefs with effective configuration class PreCompile and symbolicNameValue set to TRUE shall be generated into the header file of the declaring module as #defines. The symbol shall be composed of

- either
  - the module implementation prefix {Mip} of the declaring BSW Module
  - or the apiServicePrefix for Complex Driver Modules
  - or the apiServicePrefix for Xfrm Modules
- followed by the literal "Conf_" followed by
- the shortName of the EcucParamConfContainerDef of the declaring module followed by "_" followed by
- the shortName of the EcucContainerValue container which holds the symbolicNameValue configuration parameter value.

⌋

Taking the specification requirements above the configuration snippet results in the according symbolic name definition in the header file of the providing Dem module:

```
...
#define DemConf_DemEventParameter_CORTST_E_CORE_FAILURE_1 17
...
```

Especially in case of production error reporting this pattern is used extensively: The integrator has the freedom to call the DemEventParameter container name arbitrarily. In general it is reasonable to name the DemEventParameter like the actual production error (e.g. FLS_E_ERASE_FAILED), however there are use-cases where the same production error shall be reported for several instances / channels individually, thus it is required to distinguish between these different production error occurrences (e.g. FRIF_E_NIT_CH_A_CLUSTER_1 where FRIF_E_NIT_CH_A is the production error name and _CLUSTER_1 encodes one specific FlexRay cluster). This needs to be kept in mind when accessing the production error symbolic name from the reporting module, e.g. FrIf shall call:

```
Dem_SetEventStatus(DemConf_DemEventParameter_FRIF_E_NIT_CH_A_CLUSTER_1,
                   DEM_EVENT_STATUS_PASSED);
```

In figure 2.34 another example of a symbolic name value configuration setup is shown. The Listing 2.44 shows one possible EcuC model for this definition.

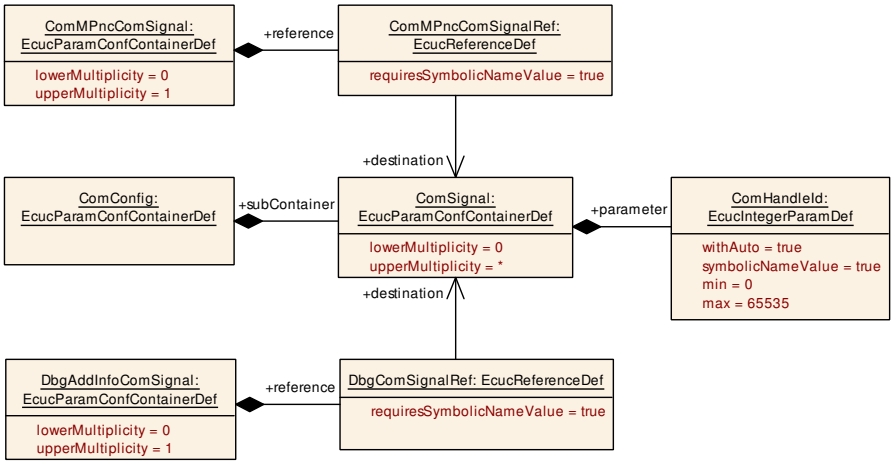

Figure 2.34: Example of ComSignal symbolic name definition

```
<ECUC-CONTAINER-VALUE>
  <SHORT-NAME>myComConfig</SHORT-NAME>
  <DEFINITION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF">/AUTOSAR/
     EcucDefs/Com/ComConfig</DEFINITION-REF>
  <SUB-CONTAINERS>
    <ECUC-CONTAINER-VALUE>
      <SHORT-NAME>PNC_02</SHORT-NAME>
      <DEFINITION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF">/AUTOSAR/
         EcucDefs/Com/ComConfig/ComSignal</DEFINITION-REF>
      <PARAMETER-VALUES>
        <ECUC-NUMERICAL-PARAM-VALUE>
          <DEFINITION-REF DEST="ECUC-INTEGER-PARAM-DEF">/AUTOSAR/
             EcucDefs/Com/ComConfig/ComSignal/ComHandleId</DEFINITION-
             REF>
          <VALUE>231</VALUE>
        </ECUC-NUMERICAL-PARAM-VALUE>
      </PARAMETER-VALUES>
    </ECUC-CONTAINER-VALUE>
    <ECUC-CONTAINER-VALUE>
```

```
        <SHORT-NAME>Debuging_Sig5</SHORT-NAME>
        <DEFINITION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF">/AUTOSAR/
            EcucDefs/Com/ComConfig/ComSignal</DEFINITION-REF>
        <PARAMETER-VALUES>
          <ECUC-NUMERICAL-PARAM-VALUE>
            <DEFINITION-REF DEST="ECUC-INTEGER-PARAM-DEF">/AUTOSAR/
                EcucDefs/Com/ComConfig/ComSignal/ComHandleId</DEFINITION-
                REF>
            <VALUE>245</VALUE>
          </ECUC-NUMERICAL-PARAM-VALUE>
        </PARAMETER-VALUES>
      </ECUC-CONTAINER-VALUE>
    </SUB-CONTAINERS>
  </ECUC-CONTAINER-VALUE>
```

Listing 2.44: Further example for Symbolc Name Reference (ComSignal)

This leads to the generation of the following definitions in the Com header file:

```
#define ComConf_ComSignal_PNC_02 231

#define ComConf_ComSignal_Debugging_Sig5 245
```

Such that the other BSW Modules - which include the Com header file - can call the Com module using these symbols:

```
ComM: Com_SendSignal(ComConf_ComSignal_PNC_02, value)

Dgb:  Com_SendSignal(ComConf_ComSignal_Debugging_Sig5, value)
```

2.4.5.2.1 Invalid configuration due to symbolic name values

[TPS_ECUC_06074] Invalid configuration due to symbolic name values ⌈Due to the hierarchical structure of the EcucParameterValues or the existence of post-build variants, it is possible that the same shortName is the base for multiple symbolicNameValue definitions. If the respective value is equal in all occurrences of the shortName according to [TPS_ECUC_02108], the generation of the #define shall only be done once. If the respective value is different in any of the occurrences of the shortName according to [TPS_ECUC_02108], the configuration is invalid.⌋

Example 2.35 shows a valid and an invalid configuration due to the existence of post-build variations.

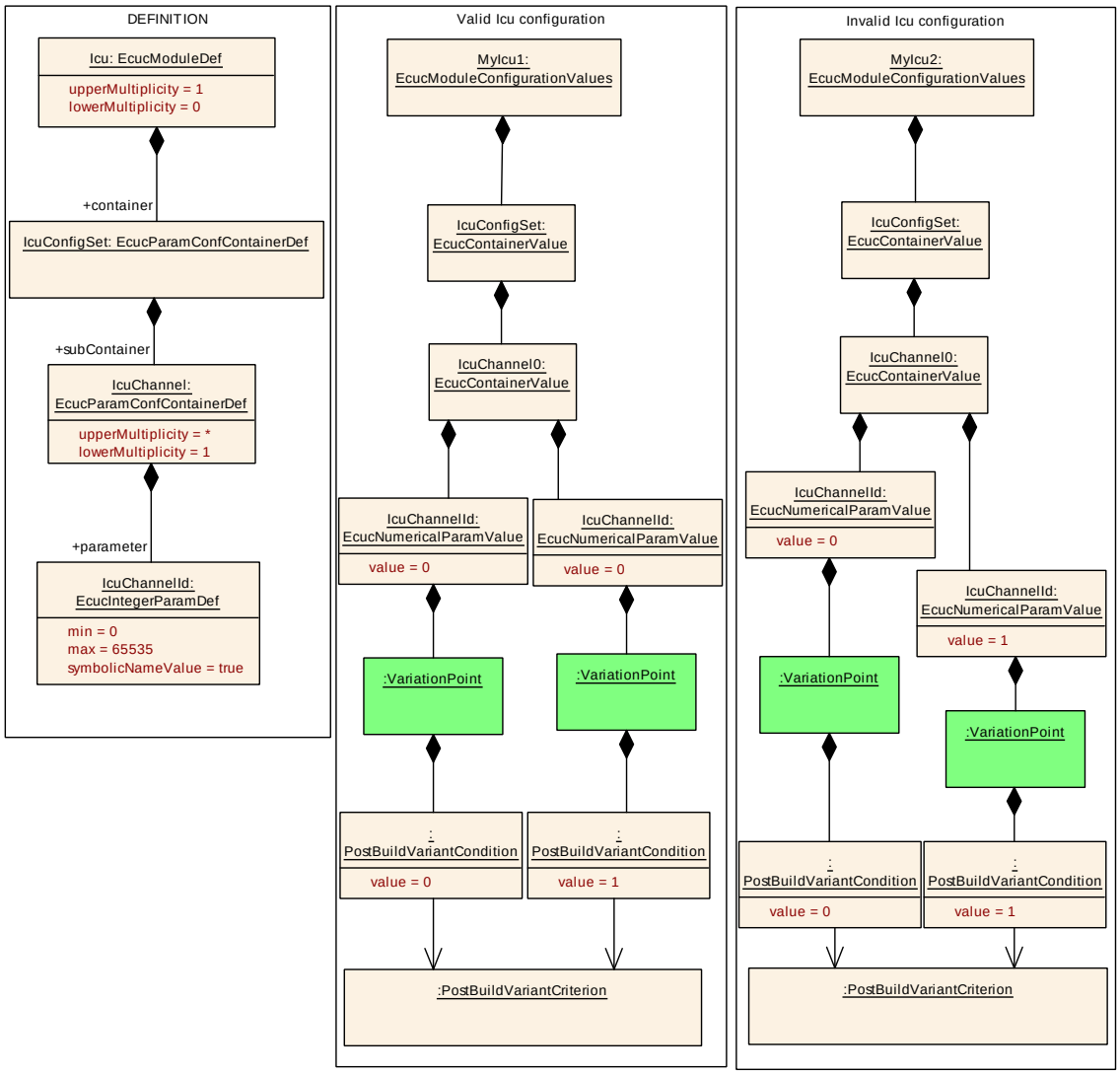

Figure 2.35: SymbolicNameValues and the generation of #defines: valid and invalid configurations due to the existence of post-build variations

The *valid* example in figure 2.35 does lead to the following definition:

```
#define IcuConf_IcuChannel_IcuChannel0 0
```

The *invalid* example in figure 2.35 would possibly lead to the following definitions:

```
#define IcuConf_IcuChannel_IcuChannel0 0

#define IcuConf_IcuChannel_IcuChannel0 1
```

where a different value would be assigned to the same symbol. This is an invalid configuration.

Example 2.36 and 2.37 shows a valid and an invalid configuration due to the hierarchical structure of the EcucParameterValues.

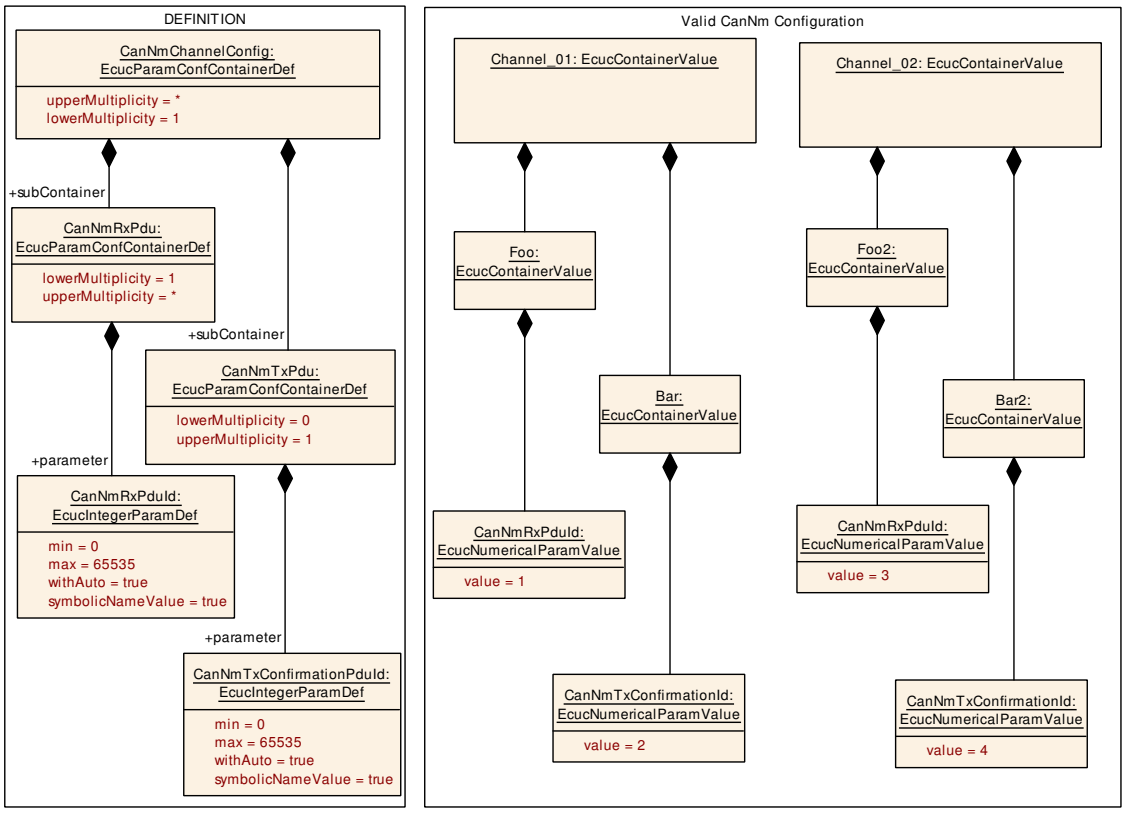

Figure 2.36: SymbolicNameValues and the generation of #defines: valid configuration due to the hierarchical structure of the EcucParameterValues

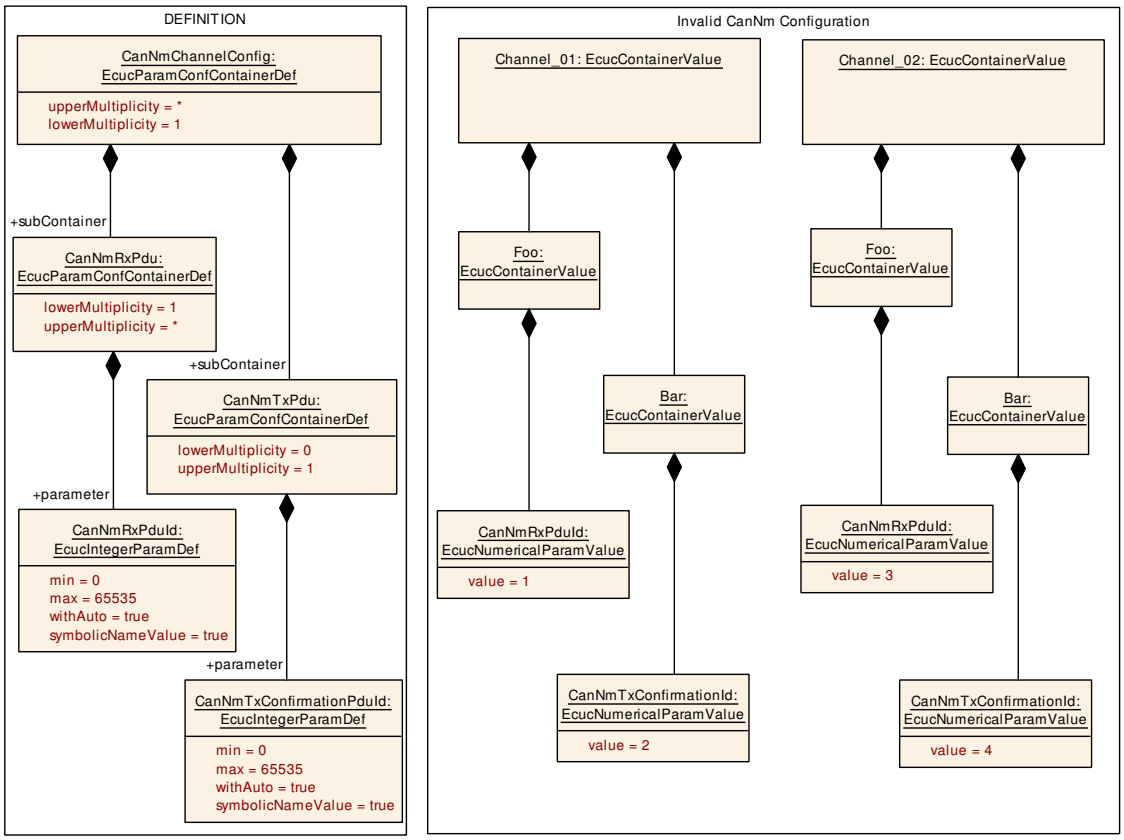

Figure 2.37: SymbolicNameValues and the generation of #defines: invalid configuration due to the hierarchical structure of the EcucParameterValues

The *valid* example in figure 2.36 does lead to the following definition:

```
#define CanNmConf_CanNmRxPdu_Foo 1

#define CanNmConf_CanNmTxPdu_Bar 2

#define CanNmConf_CanNmRxPdu_Foo2 3

#define CanNmConf_CanNmTxPdu_Bar2 4
```

The *invalid* example in figure 2.37 would possibly lead to the following definitions:

```
#define CanNmConf_CanNmRxPdu_Foo 1

#define CanNmConf_CanNmTxPdu_Bar 2

#define CanNmConf_CanNmRxPdu_Foo 3

#define CanNmConf_CanNmTxPdu_Bar 4
```

where different values would be assigned to the same symbol. The value 1 would be redefined to 3 and the value 2 would be redefined to 4. This is an invalid configuration.

2.4.5.3 Rules for references in the ECUC Configuration Values

[TPS_ECUC_06047] References in the `ECU Configuration Values` with reference definitions that refer to container definitions in the same module definition
⌈For reference definitions that refer to container definitions in the same module definition the references on the value side shall only refer to container instances of this module instance.⌋

The example in figure 2.38 defines a reference inside the CanDrv module. Thus the values can only refer to container instances within the respective CanDrv configuration instance.

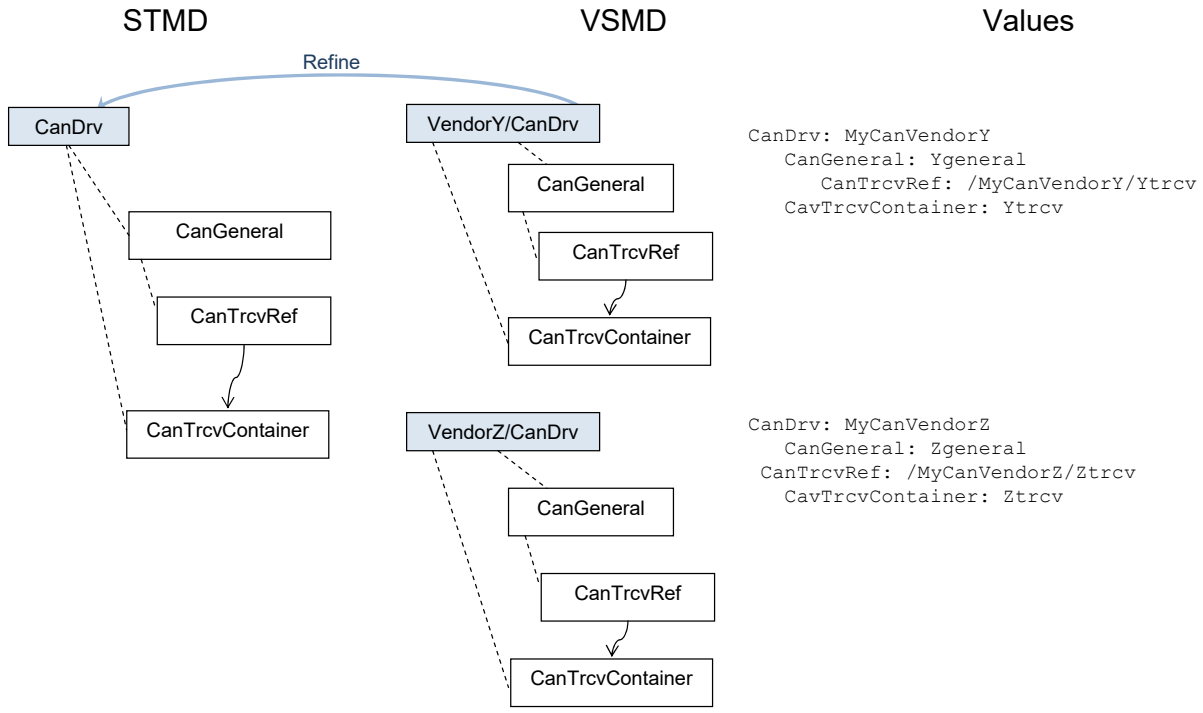

Figure 2.38: Reference inside a module

[TPS_ECUC_06048] References in the `ECU Configuration Values` with reference definitions that refer to container definitions in different module definitions
⌈For reference definitions that refer to container definitions in a different module definition the references on the value side may refer to container instances of different module instances according to the same module definition.⌋

The example in figure 2.39 defines a reference between the CanIf and the CanDrv module. Thus the values can refer to container instances of different CanDrv configuration instances.

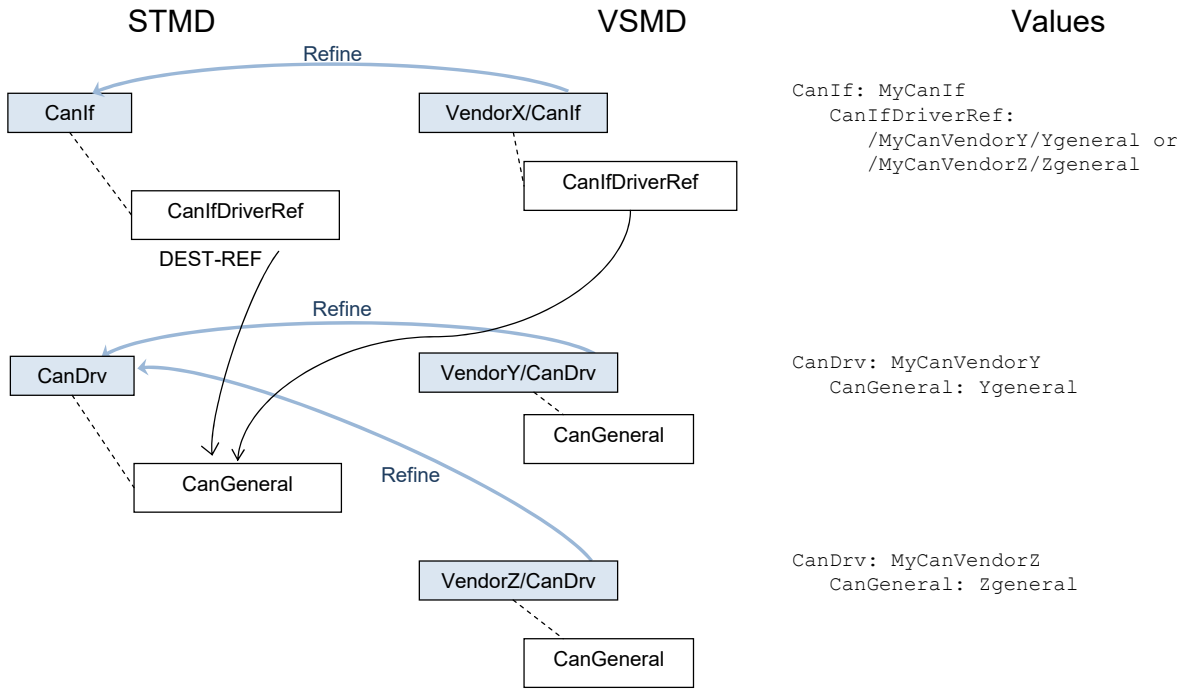

Figure 2.39: Reference between modules

2.4.6 Derived Parameters in ECU Configuration Values

[TPS_ECUC_03021] **EcucParameterDefs with EcucDerivationSpecification result in a EcucNumericalParamValue in the ECU Configuration Values** ⌈ Providing the configuration value for an instance of an EcucParameterDef which has as EcucDerivationSpecification results in a EcucNumericalParamValue.⌋

[TPS_ECUC_02125] Value of parameters with a defined derivation specification ⌈ The value of a parameter shall be provided even when the defining EcucParameterDef has a EcucDerivationSpecification.⌋

In this way it is guaranteed that even when a tool does not support the derivation formula the value is still available.

With the storage of the value it is also possible to implement consistency checks whether the actually provided value matches the result of the derivation formula.

Listing 2.45 depicts the ECU Configuration Values ARXML of derived parameters for Listing 2.21.

```
<ECUC-NUMERICAL-PARAM-VALUE>
  <DEFINITION-REF DEST="ECUC-BOOLEAN-PARAM-DEF">/AUTOSAR/EcucDefs/Com/
     ComConfig/ComGwMapping/CheckConsistency</DEFINITION-REF>
  <VALUE>1</VALUE>
```

```
</ECUC-NUMERICAL-PARAM-VALUE>
```

Listing 2.45: Derived Value Example

2.4.7 Using Variant Handling to Cope with Several Binding Times in the ECU Configuration Values

The goal of this feature is to provide modeling support to handle several binding times of the ECU Configuration Values in one model. The idea is to utilize Variant Handling approach to allow different values and/or different number of instances of certain configuration parameters in different variation points bound at post-build time (referred to as post-build variants). In order to achieve this, at least one PostBuildVariantCriterion shall be declared in order to define a common selecting element for different post-build variants. The variants are specified using different PostBuildVariantCriterionValues. An example of one criterion with two values is shown in 2.46:

```
<POST-BUILD-VARIANT-CRITERION>
  <SHORT-NAME>PostBuildConfigSet</SHORT-NAME>
</POST-BUILD-VARIANT-CRITERION>
<POST-BUILD-VARIANT-CRITERION-VALUE-SET>
  <SHORT-NAME>PostBuildVariants</SHORT-NAME>
  <POST-BUILD-VARIANT-CRITERION-VALUES>
    <POST-BUILD-VARIANT-CRITERION-VALUE>
      <VARIANT-CRITERION-REF DEST="POST-BUILD-VARIANT-CRITERION">/EcucDemo/
          PostBuildConfigSet</VARIANT-CRITERION-REF>
      <VALUE>1</VALUE>
    </POST-BUILD-VARIANT-CRITERION-VALUE>
    <POST-BUILD-VARIANT-CRITERION-VALUE>
      <VARIANT-CRITERION-REF DEST="POST-BUILD-VARIANT-CRITERION">/EcucDemo/
          PostBuildConfigSet</VARIANT-CRITERION-REF>
      <VALUE>2</VALUE>
    </POST-BUILD-VARIANT-CRITERION-VALUE>
  </POST-BUILD-VARIANT-CRITERION-VALUES>
</POST-BUILD-VARIANT-CRITERION-VALUE-SET>
```

Listing 2.46: Example for one PostBuildVariantCriterion with two PostBuildVariantCriterionValues

[TPS_ECUC_08011] Pattern for creating a C symbol used by the EcuM/BswM to initialize BSW modules with different post-build variants ⌈For the name mangling of symbols of different post-build variants (configuration sets) of one BSW module, the following pattern shall be used:

<Mip>_ConfigType <Mip>_Config[_<PredefinedVariant.shortName>]

where <Mip> is the module implementation prefix according to [SWS_BSW_00102], <PredefinedVariant.shortName> is the shortName of the PredefinedVariant referenced by EcucPostBuildVariantRef reference in the EcucPostBuildVariants container of the respective module.

In case of pure post-build configuration without post-build variants, the optional suffix _<PredefinedVariant.shortName> shall be omitted.⌋

2.4.7.1 Example of ECU configuration using `Variant Handling`

This section contains an example of how ECU configuration parameters with `postBuildVariantValue` and `postBuildVariantMultiplicity` attributes set to `true` or `false` inside containers with `postBuildVariantMultiplicity` attribute set to `true` / `false` can be configured using `Variant Handling`. As an example, a part of the Adc module configuration parameters is taken (see Figure 2.40).

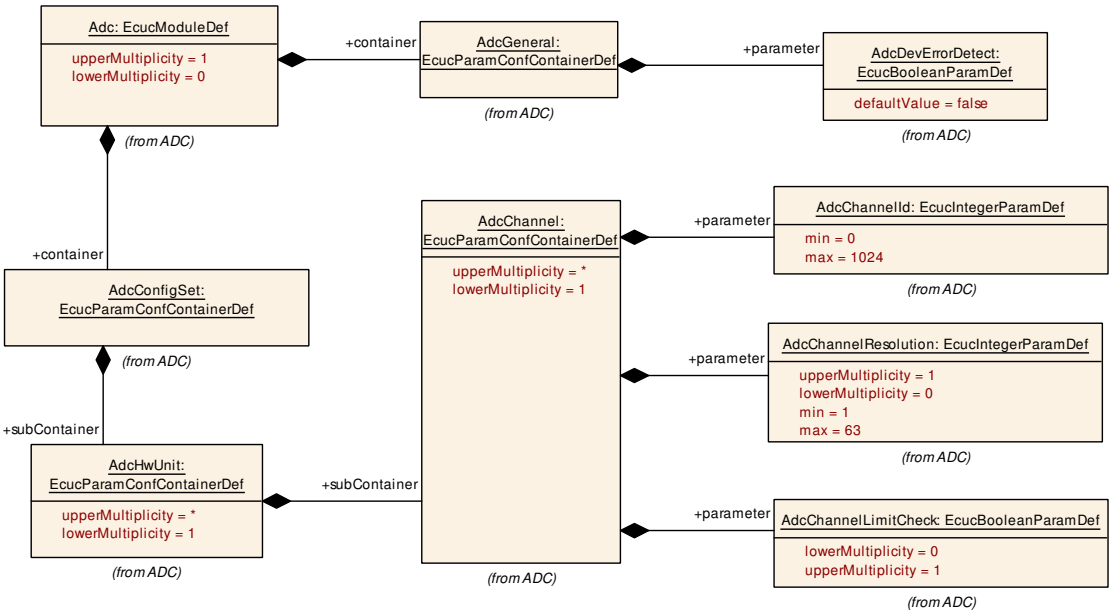

Figure 2.40: Example of Parameters Configuration Using Variant Handling

The `AdcDevErrorDetect` parameter of the `AdcGeneral` container and the `AdcChannelLimitCheck` parameter of the `AdcChannel` container shall have the same value in all post-build variants (i.e. `postBuildVariantValue` is set to `false`) while the `AdcChannelId` and the `AdcChannelResolution` parameters of the `AdcChannel` container can have different values in different post-build variants (i.e. `postBuildVariantValue` is set to `true`). All parameters shall have the same number of instances in different post-build variants (i.e. `postBuildVariantMultiplicity` is set to `false`). The container `AdcGeneral` cannot change its number of instances between different variants (i.e. `postBuildVariantMultiplicity` attribute set to `false`) while the container `AdcChannel` can (i.e. `postBuildVariantMultiplicity` attribute set to `true`). This is depicted in Listing 2.47.

```
<ECUC-MODULE-DEF>
  <SHORT-NAME>Adc</SHORT-NAME>
  <LOWER-MULTIPLICITY>0</LOWER-MULTIPLICITY>
  <UPPER-MULTIPLICITY>1</UPPER-MULTIPLICITY>
  <POST-BUILD-LOADABLE-SUPPORT>true</POST-BUILD-LOADABLE-SUPPORT>
  <POST-BUILD-VARIANT-SUPPORT>true</POST-BUILD-VARIANT-SUPPORT>
  <CONTAINERS>
    <!-- Container Definition: AdcConfigSet -->
    <ECUC-PARAM-CONF-CONTAINER-DEF>
      <SHORT-NAME>AdcConfigSet</SHORT-NAME>
      <LOWER-MULTIPLICITY>1</LOWER-MULTIPLICITY>
```

```
<UPPER-MULTIPLICITY>1</UPPER-MULTIPLICITY>
<SUB-CONTAINERS>
  <!-- Container Definition: AdcHwUnit -->
  <ECUC-PARAM-CONF-CONTAINER-DEF>
    <SHORT-NAME>AdcHwUnit</SHORT-NAME>
    <LOWER-MULTIPLICITY>1</LOWER-MULTIPLICITY>
    <UPPER-MULTIPLICITY-INFINITE>true</UPPER-MULTIPLICITY-INFINITE>
    <POST-BUILD-LOADABLE-MULTIPLICITY>false</POST-BUILD-LOADABLE-
       MULTIPLICITY>
    <POST-BUILD-VARIANT-MULTIPLICITY>false</POST-BUILD-VARIANT-
       MULTIPLICITY>
    <PARAMETERS/>
    <SUB-CONTAINERS>
      <!-- Container Definition: AdcChannel -->
      <ECUC-PARAM-CONF-CONTAINER-DEF>
        <SHORT-NAME>AdcChannel</SHORT-NAME>
        <LOWER-MULTIPLICITY>1</LOWER-MULTIPLICITY>
        <UPPER-MULTIPLICITY-INFINITE>true</UPPER-MULTIPLICITY-
           INFINITE>
        <POST-BUILD-LOADABLE-MULTIPLICITY>true</POST-BUILD-LOADABLE-
           MULTIPLICITY>
        <POST-BUILD-VARIANT-MULTIPLICITY>true</POST-BUILD-VARIANT-
           MULTIPLICITY>
        <PARAMETERS>
          <!-- PARAMETER DEFINITION: AdcChannelId -->
          <ECUC-INTEGER-PARAM-DEF>
            <SHORT-NAME>AdcChannelId</SHORT-NAME>
            <LOWER-MULTIPLICITY>1</LOWER-MULTIPLICITY>
            <UPPER-MULTIPLICITY>1</UPPER-MULTIPLICITY>
            <ORIGIN>AUTOSAR_ECUC</ORIGIN>
            <POST-BUILD-LOADABLE-VALUE>true</POST-BUILD-LOADABLE-
               VALUE>
            <POST-BUILD-VARIANT-VALUE>true</POST-BUILD-VARIANT-VALUE>
            <SYMBOLIC-NAME-VALUE>false</SYMBOLIC-NAME-VALUE>
            <MAX>1024</MAX>
            <MIN>0</MIN>
          </ECUC-INTEGER-PARAM-DEF>
          <!-- PARAMETER DEFINITION: AdcChannelLimitCheck -->
          <ECUC-BOOLEAN-PARAM-DEF>
            <SHORT-NAME>AdcChannelLimitCheck</SHORT-NAME>
            <DESC>
              <L-2 L="EN">Enables or disables limit checking for an
                 ADC channel.</L-2>
            </DESC>
            <LOWER-MULTIPLICITY>0</LOWER-MULTIPLICITY>
            <UPPER-MULTIPLICITY>1</UPPER-MULTIPLICITY>
            <ORIGIN>AUTOSAR_ECUC</ORIGIN>
            <POST-BUILD-LOADABLE-MULTIPLICITY>true</POST-BUILD-
               LOADABLE-MULTIPLICITY>
            <POST-BUILD-LOADABLE-VALUE>true</POST-BUILD-LOADABLE-
               VALUE>
            <POST-BUILD-VARIANT-MULTIPLICITY>false</POST-BUILD-
               VARIANT-MULTIPLICITY>
            <POST-BUILD-VARIANT-VALUE>false</POST-BUILD-VARIANT-VALUE
               >
            <SYMBOLIC-NAME-VALUE>false</SYMBOLIC-NAME-VALUE>
```

```
                </ECUC-BOOLEAN-PARAM-DEF>
                <!-- PARAMETER DEFINITION: AdcChannelResolution -->
                <ECUC-INTEGER-PARAM-DEF>
                  <SHORT-NAME>AdcChannelResolution</SHORT-NAME>
                  <LOWER-MULTIPLICITY>0</LOWER-MULTIPLICITY>
                  <UPPER-MULTIPLICITY>1</UPPER-MULTIPLICITY>
                  <ORIGIN>AUTOSAR_ECUC</ORIGIN>
                  <POST-BUILD-LOADABLE-MULTIPLICITY>false</POST-BUILD-
                      LOADABLE-MULTIPLICITY>
                  <POST-BUILD-LOADABLE-VALUE>true</POST-BUILD-LOADABLE-
                      VALUE>
                  <POST-BUILD-VARIANT-MULTIPLICITY>false</POST-BUILD-
                      VARIANT-MULTIPLICITY>
                  <POST-BUILD-VARIANT-VALUE>true</POST-BUILD-VARIANT-VALUE>
                  <SYMBOLIC-NAME-VALUE>false</SYMBOLIC-NAME-VALUE>
                  <MAX>63</MAX>
                  <MIN>1</MIN>
                </ECUC-INTEGER-PARAM-DEF>
              </PARAMETERS>
            </ECUC-PARAM-CONF-CONTAINER-DEF>
          </SUB-CONTAINERS>
        </ECUC-PARAM-CONF-CONTAINER-DEF>
      </SUB-CONTAINERS>
    </ECUC-PARAM-CONF-CONTAINER-DEF>
    <!-- Container Definition: AdcGeneral -->
    <ECUC-PARAM-CONF-CONTAINER-DEF>
      <SHORT-NAME>AdcGeneral</SHORT-NAME>
      <LOWER-MULTIPLICITY>1</LOWER-MULTIPLICITY>
      <UPPER-MULTIPLICITY>1</UPPER-MULTIPLICITY>
      <PARAMETERS>
        <!-- PARAMETER DEFINITION: AdcDevErrorDetect -->
        <ECUC-BOOLEAN-PARAM-DEF>
          <SHORT-NAME>AdcDevErrorDetect</SHORT-NAME>
          <LOWER-MULTIPLICITY>1</LOWER-MULTIPLICITY>
          <UPPER-MULTIPLICITY>1</UPPER-MULTIPLICITY>
          <ORIGIN>AUTOSAR_ECUC</ORIGIN>
          <POST-BUILD-LOADABLE-VALUE>false</POST-BUILD-LOADABLE-VALUE>
          <POST-BUILD-VARIANT-VALUE>false</POST-BUILD-VARIANT-VALUE>
          <SYMBOLIC-NAME-VALUE>false</SYMBOLIC-NAME-VALUE>
        </ECUC-BOOLEAN-PARAM-DEF>
      </PARAMETERS>
    </ECUC-PARAM-CONF-CONTAINER-DEF>
  </CONTAINERS>
</ECUC-MODULE-DEF>
```

Listing 2.47: Parameter Definition Example for Variant Handling

The parameters with fixed value in all post-build variants inside containers with fixed number of instances are provided normally inside the container structure they are defined in (see `AdcDevErrorDetect` parameter in Listing 2.48).

```
<ECUC-MODULE-CONFIGURATION-VALUES>
  <SHORT-NAME>theAdc</SHORT-NAME>
  <DEFINITION-REF DEST="ECUC-MODULE-DEF">/EcucDemo/Adc</DEFINITION-REF>
  <CONTAINERS>
    <ECUC-CONTAINER-VALUE>
```

```
      <SHORT-NAME>myGeneralValue</SHORT-NAME>
      <DEFINITION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF">/EcucDemo/Adc/
          AdcGeneral</DEFINITION-REF>
      <PARAMETER-VALUES>
        <ECUC-NUMERICAL-PARAM-VALUE>
          <DEFINITION-REF DEST="ECUC-BOOLEAN-PARAM-DEF">/EcucDemo/Adc/
              AdcGeneral/AdcDevErrorDetect</DEFINITION-REF>
          <VALUE>1</VALUE>
        </ECUC-NUMERICAL-PARAM-VALUE>
      </PARAMETER-VALUES>
      <SUB-CONTAINERS/>
    </ECUC-CONTAINER-VALUE>
  </CONTAINERS>
</ECUC-MODULE-CONFIGURATION-VALUES>
        <!--  -->
```

Listing 2.48: Parameter Values without variation (fixed values) in fixed containers

Similarly, the parameters with fixed value in all post-build variants inside containers with possible different number of instances in different variants (i.e. `postBuildVariantMultiplicity` is set to `true`) also do not need to be duplicated in every variant. Instead they should be defined only once which also guarantees that the value of these parameters are the same in all variants (see `AdcChannelLimitCheck` parameter in Listing 2.49).

```
<ECUC-MODULE-CONFIGURATION-VALUES>
  <SHORT-NAME>theAdc</SHORT-NAME>
  <DEFINITION-REF DEST="ECUC-MODULE-DEF">/EcucDemo/Adc</DEFINITION-REF>
  <CONTAINERS>
    <ECUC-CONTAINER-VALUE>
      <SHORT-NAME>myHwUnit</SHORT-NAME>
      <DEFINITION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF">/EcucDemo/Adc/
          AdcConfigSet/AdcHwUnit</DEFINITION-REF>
      <SUB-CONTAINERS>
        <ECUC-CONTAINER-VALUE>
          <SHORT-NAME>myChannel1</SHORT-NAME>
          <DEFINITION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF">/EcucDemo/
              Adc/AdcConfigSet/AdcHwUnit/AdcChannel</DEFINITION-REF>
          <PARAMETER-VALUES>
            <ECUC-NUMERICAL-PARAM-VALUE>
              <DEFINITION-REF DEST="ECUC-BOOLEAN-PARAM-DEF">/EcucDemo/Adc/
                  AdcConfigSet/AdcHwUnit/AdcChannel/AdcChannelLimitCheck</
                  DEFINITION-REF>
              <VALUE>0</VALUE>
            </ECUC-NUMERICAL-PARAM-VALUE>
          </PARAMETER-VALUES>
        </ECUC-CONTAINER-VALUE>
      </SUB-CONTAINERS>
    </ECUC-CONTAINER-VALUE>
  </CONTAINERS>
</ECUC-MODULE-CONFIGURATION-VALUES>
```

Listing 2.49: Parmeter Values without variation (fixed values) in variant containers

However for parameters which may have different value in different post-build variants (i.e. postBuildVariantValue is set to true), the PostBuildVariantCriterion shall be referenced in order to define the common selector. A specific value for the selector is defined for each post-build variant to specify to which variant this parameter value is associated to (see AdcChannelResolution parameter in two post-build variants, left and right, in Listing 2.50)..

```
<ECUC-MODULE-CONFIGURATION-VALUES>
  <SHORT-NAME>theAdc</SHORT-NAME>
  <DEFINITION-REF DEST="ECUC-MODULE-DEF">/EcucDemo/Adc</DEFINITION-REF>
  <CONTAINERS>
    <ECUC-CONTAINER-VALUE>
      <SHORT-NAME>myHwUnit</SHORT-NAME>
      <DEFINITION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF">/EcucDemo/Adc/
          AdcConfigSet/AdcHwUnit</DEFINITION-REF>
      <SUB-CONTAINERS>
        <ECUC-CONTAINER-VALUE>
          <SHORT-NAME>myChannel1</SHORT-NAME>
          <DEFINITION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF">/EcucDemo/
              Adc/AdcConfigSet/AdcHwUnit/AdcChannel</DEFINITION-REF>
          <PARAMETER-VALUES>
            <ECUC-NUMERICAL-PARAM-VALUE>
              <DEFINITION-REF DEST="ECUC-INTEGER-PARAM-DEF">/EcucDemo/Adc/
                  AdcConfigSet/AdcHwUnit/AdcChannel/AdcChannelResolution</
                  DEFINITION-REF>
              <VARIATION-POINT>
                <POST-BUILD-VARIANT-CONDITIONS>
                  <POST-BUILD-VARIANT-CONDITION>
                    <MATCHING-CRITERION-REF DEST="POST-BUILD-VARIANT-
                        CRITERION">/EcucDemo/PostBuildConfigSet</MATCHING-
                        CRITERION-REF>
                    <VALUE>1</VALUE>
                    <!-- PostBuildFrontLeft -->
                  </POST-BUILD-VARIANT-CONDITION>
                </POST-BUILD-VARIANT-CONDITIONS>
              </VARIATION-POINT>
              <VALUE>10</VALUE>
            </ECUC-NUMERICAL-PARAM-VALUE>
            <ECUC-NUMERICAL-PARAM-VALUE>
              <DEFINITION-REF DEST="ECUC-INTEGER-PARAM-DEF">/EcucDemo/Adc/
                  AdcConfigSet/AdcHwUnit/AdcChannel/AdcChannelResolution</
                  DEFINITION-REF>
              <VARIATION-POINT>
                <POST-BUILD-VARIANT-CONDITIONS>
                  <POST-BUILD-VARIANT-CONDITION>
                    <MATCHING-CRITERION-REF DEST="POST-BUILD-VARIANT-
                        CRITERION">/EcucDemo/PostBuildConfigSet</MATCHING-
                        CRITERION-REF>
                    <VALUE>2</VALUE>
                    <!-- PostBuildFrontRight -->
                  </POST-BUILD-VARIANT-CONDITION>
                </POST-BUILD-VARIANT-CONDITIONS>
              </VARIATION-POINT>
              <VALUE>40</VALUE>
            </ECUC-NUMERICAL-PARAM-VALUE>
```

```
          </PARAMETER-VALUES>
        </ECUC-CONTAINER-VALUE>
      </SUB-CONTAINERS>
    </ECUC-CONTAINER-VALUE>
  </CONTAINERS>
</ECUC-MODULE-CONFIGURATION-VALUES>
```

Listing 2.50: Parmeter Values with variation (varying values)

If one container shall be used in all post-build variants (e.g. because there are pre-compile configurations pointing to this container), it shall not define a variation point and thus indicate its "pre-compile" nature. In case a container exists only in some post-build variants, it shall define a variation point. Also all included elements shall then define the respective variation point (see `AdcChannel` container in Listing 2.51).

```
<ECUC-MODULE-CONFIGURATION-VALUES>
  <SHORT-NAME>theAdc</SHORT-NAME>
  <DEFINITION-REF DEST="ECUC-MODULE-DEF">/EcucDemo/Adc</DEFINITION-REF>
  <CONTAINERS>
    <ECUC-CONTAINER-VALUE>
      <SHORT-NAME>myChannel5</SHORT-NAME>
      <DEFINITION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF">/EcucDemo/Adc/
          AdcConfigSet/AdcHwUnit/AdcChannel</DEFINITION-REF>
      <PARAMETER-VALUES>
        <ECUC-NUMERICAL-PARAM-VALUE>
          <DEFINITION-REF DEST="ECUC-BOOLEAN-PARAM-DEF">/EcucDemo/Adc/
              AdcConfigSet/AdcHwUnit/AdcChannel/AdcChannelLimitCheck</
              DEFINITION-REF>
          <VALUE>1</VALUE>
        </ECUC-NUMERICAL-PARAM-VALUE>
        <ECUC-NUMERICAL-PARAM-VALUE>
          <DEFINITION-REF DEST="ECUC-INTEGER-PARAM-DEF">/EcucDemo/Adc/
              AdcConfigSet/AdcHwUnit/AdcChannel/AdcChannelId</DEFINITION-REF
              >
          <VALUE>5</VALUE>
        </ECUC-NUMERICAL-PARAM-VALUE>
        <ECUC-NUMERICAL-PARAM-VALUE>
          <DEFINITION-REF DEST="ECUC-INTEGER-PARAM-DEF">/EcucDemo/Adc/
              AdcConfigSet/AdcHwUnit/AdcChannel/AdcChannelResolution</
              DEFINITION-REF>
          <VARIATION-POINT>
            <POST-BUILD-VARIANT-CONDITIONS>
              <POST-BUILD-VARIANT-CONDITION>
                <MATCHING-CRITERION-REF DEST="POST-BUILD-VARIANT-CRITERION"
                    >/EcucDemo/PostBuildConfigSet</MATCHING-CRITERION-REF>
                <VALUE>2</VALUE>
                <!-- PostBuildFrontRight -->
              </POST-BUILD-VARIANT-CONDITION>
            </POST-BUILD-VARIANT-CONDITIONS>
          </VARIATION-POINT>
          <VALUE>30</VALUE>
        </ECUC-NUMERICAL-PARAM-VALUE>
      </PARAMETER-VALUES>
      <VARIATION-POINT>
        <POST-BUILD-VARIANT-CONDITIONS>
          <POST-BUILD-VARIANT-CONDITION>
            <MATCHING-CRITERION-REF DEST="POST-BUILD-VARIANT-CRITERION">/
                EcucDemo/PostBuildConfigSet</MATCHING-CRITERION-REF>
            <VALUE>2</VALUE>
            <!-- PostBuildFrontRight -->
          </POST-BUILD-VARIANT-CONDITION>
        </POST-BUILD-VARIANT-CONDITIONS>
      </VARIATION-POINT>
    </ECUC-CONTAINER-VALUE>
  </CONTAINERS>
</ECUC-MODULE-CONFIGURATION-VALUES>
```

Listing 2.51: Example for an `EcucContainerValue` that only exists in some variants (`myChannel5` only exists in `PostBuildConfigSet` 2)

3 ECU Configuration Parameter Definition SWS implications

In this section several aspects of applying the ECU Configuration Specification to AUTOSAR specifications are described.

The ECU Configuration Parameter Definitions are distributed over the BSW SWS documents. How these parameters are specified in the documents is described in section 3.1.

How the AUTOSAR COM-Stack is configured from an inter-module perspective is described in section 3.4.

3.1 Formalization aspects

The goal of this section is to describe how the ECU Configuration Parameter Definitions of BSW modules are specified in the SWS documents. There is not necessarily a simple translation of the ECU Configuration Parameter's values in the ECU Configuration Values ARXML into the module's configuration (header file). It is the duty of the module's generation tool to transform the configuration information from the XML file into a header file.

The ECU Configuration Parameter Definitions are formalized in an UML model. This UML model is used to partly generate the specification tables of the BSW SWS and to generate the ECU Configuration Parameter Definition XML file. [1]

Some formalization patterns have been applied when developing the ECU Configuration Parameter Definition:

- *Modified parameter names*: Due to the limitations imposed by the AUTOSAR XML format (32 character limit starting with a letter, etc.) the names of parameters and containers have been redefined. Also a different naming schema has been applied. The original names from the SWS are provided in this document as well.
- *Added parameter multiplicities*: In the original tables from the BSW SWS there is no possibility to specify the optionality and multiplicity of parameters. The parameter multiplicities have been added.
- *Added references*: To allow a better interaction of the configuration Values of several modules references between the configuration have been introduced.
- *Harmonized parameter types*:

[1] The generation from the UML model is only one way to create the ECU Configuration Parameter Definition XML file. ECUC Parameters can be defined by any other method as long as an AUTOSAR conforming EcuC Parameter Definition XML file is created.

– Boolean: Some parameters have been defined as `enumeration` or `#define` where the actual information stored is of type boolean. In those cases they have been modeled as `boolean`.

– Float: Some parameters store a time value as `integer` where it is stated that this is a time in e.g. micro-seconds. If the time specified is an absolute time it has been formalized as a `float` in seconds. If the time is a factor of some given time-base the `integer` is preserved.

3.1.1 ECU Configuration Parameter Definition table

The configuration parameters are structured into containers which can hold parameters, references and other containers. Beside the graphical visualization in UML diagrams, tables are used to specify the structure of the parameters.

In the following table one container is specified which holds two parameters and also two additional containers as an example.

SWS Item	**[ECUC_MA_00001]**	
Container Name	ContainerName	
Parent Container	aggregatingContainerName	
Description	Container description	
Post-Build Multiplicity	Loadable	Selectable
Configuration Parameters		

SWS Item	**[ECUC_MA_00002]**	
Parameter Name	ParameterName1	
Parent Container	ContainerName	
Description	Parameter description	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 4294967295	
Default value	0	
Post-Build Value	Loadable	Selectable

SWS Item	**[ECUC_MA_00003]**	
Parameter Name	ParameterName2	
Parent Container	ContainerName	
Description	Parameter description	
Multiplicity	0..1	
Type	EcucIntegerParamDef	
Range	0 .. 64	
Default value		
Post-Build Multiplicity	Loadable	-
Post-Build Value	Loadable	Selectable

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
Container1	0..1	Optional sub-container
Container2	0..*	Optional sub-container

For a detailed description of the elements in the tables please refer to chapter 2.

3.2 AUTOSAR Stack Overview

The software architecture of an AUTOSAR ECU has been divided into several parts to allow independent modules with clean definitions of the interfaces between the different modules. This architecture is depicted in figure 3.1.

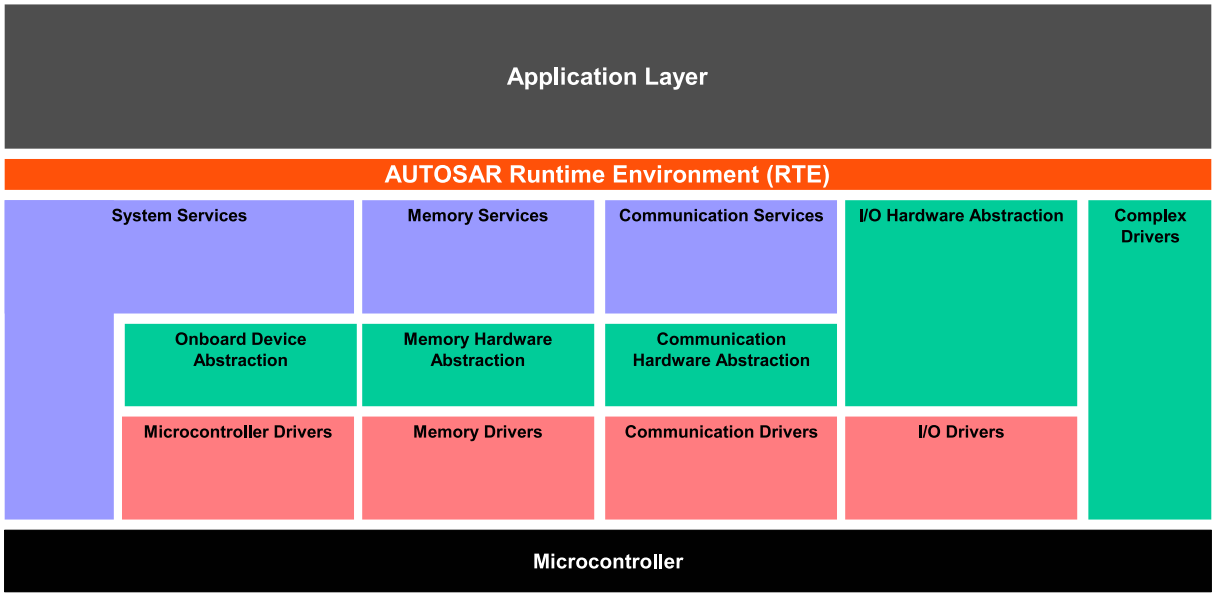

Figure 3.1: ECU Architecture Overview [12]

The Application SW-Components are located at the top and can gain access to the rest of the ECU and also to other ECUs only through the RTE.

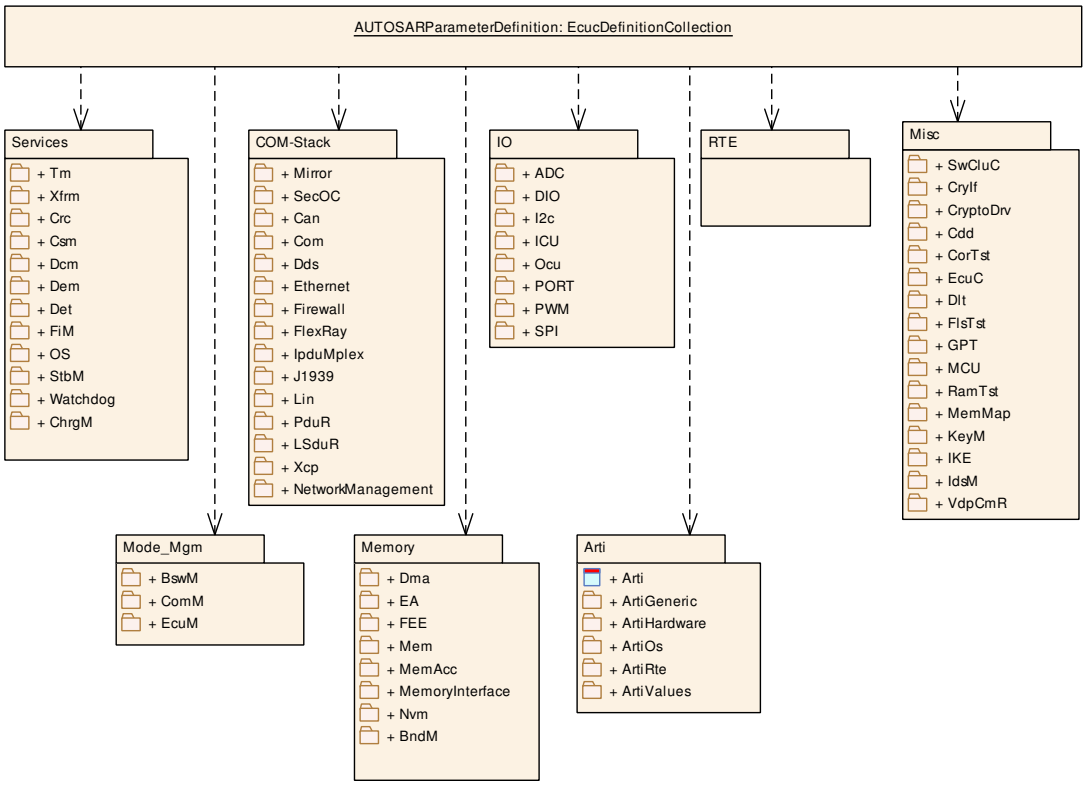

Figure 3.2: AUTOSAR Parameter Definition Overview

The RTE provides the encapsulation of communication and basic services to the Application SW-Components, so it is possible to map the Application SW-Components between different ECUs.

The Basic Software Modules are located below the RTE. The Basic Software itself is divided into the subgroups: System Services, Memory, Communication and IO HW-Abstraction. The Complex Drivers are also located below the RTE.

Among other, the Operating System (OS), the Watchdog manager and the Diagnostic services are located in the System Services subgroup.

The Memory subgroup contains modules to provide access to the non-volatile memories, namely Flash and EEPROM.

In the Communication subgroup the whole AUTOSAR communication stack (COM-Stack) is specified including the COM, Network Management and the communication drivers.

The top-level structure of the `AUTOSARParameterDefinition` is shown in figure 3.2.

The container `AUTOSARParameterDefinition` is the top-level element of the AUTOSAR `ECU Configuration Parameter Definition` structure. Inside this container references to the diverse configuration container definitions for the different SW modules are defined.

The upper multiplicities defined in the context of each `EcucModuleDef` directly impact the instantiation of the specific modules. If `EcucModuleDef.upperMultiplicity` is

set to 1 this means that the respective module can only appear once in an AUTOSAR BSW stack. If the value of `EcucModuleDef.upperMultiplicity` is greater than 1 (i.e. 0..*) the module can be multiply instantiated.

[ECUC_EcuC_00084] Definition of EcucDefinitionCollection AUTOSARParameter Definition ⌈

ECU Conf. Name	AUTOSARParameterDefinition
Description	Top level container for the definition of AUTOSAR configuration parameters. All of the parameter definitions for the different modules are contained in this container.

Included Modules		
Module Name	**Multiplicity**	**Description**
Adc	0..1	Configuration of the Adc (Analog Digital Conversion) module.
Arti	0..1	The Arti Module serves as a superordinate container collecting all information and parameters concerning ARTI.
BndM	0..1	Configuration of the BulkNvDataManager module.
BswM	0..1	Configuration of the BswM (Basic SW Mode Manager) module.
Can	0..*	This container holds the configuration of a single CAN Driver.
CanIf	0..1	This container includes all necessary configuration sub-containers according the CAN Interface configuration structure.
CanNm	0..1	Configuration Parameters for the Can Nm module.
CanSM	0..1	Configuration of the CanSM module
CanTSyn	0..1	Configuration of the Synchronized Time-base Manager (StbM) module with respect to global time handling on CAN.
CanTp	0..1	Configuration of the CanTp (CAN Transport Protocol) module.
CanTrcv	0..*	Configuration of the CanTrcv (CAN Transceiver driver) module.
Cdd	0..*	The CDD module describes the minimal requirements that are necessary for the configuration of a CDD with respect to the surrounding standardized BSW modules.
Com	0..1	Configuration of the AUTOSAR COM module.
ComM	0..1	Configuration of the ComM (Communications Manager) module.
CorTst	0..1	Configuration of the CorTst module.
Crc	0..1	Configuration of the Crc (Crc routines) module.
CryIf	0..1	Configuration of the Crypto Interface.
Crypto	0..*	Configuration of the Crypto (CryptoDriver) module
Csm	0..1	Configuration of the Csm (CryptoServiceManager) module.
Dcm	0..1	Configuration of the Dcm (Diagnostic Communications Manager) module.
Dds	0..1	Configuration of the Dds module.
Dem	0..1	Configuration of the Dem (Diagnostic Event Manager) module.
Det	0..1	Det configuration includes the functions to be called at notification.
Dio	0..1	Configuration of the Dio (Digital IO) module.
Dlt	0..1	Configuration of the Dlt (Log&Trace) module.
Dma	0..1	Configuration of the Dma (Direct Memory Access) Driver module.
DoIP	0..1	Configuration of the DoIP (Diagnostic over IP) module.

▽

△

Included Modules		
Module Name	**Multiplicity**	**Description**
Ea	0..1	Configuration of the Ea (EEPROM Abstraction) module. The module shall abstract from the device specific addressing scheme and segmentation and provide the upper layers with a virtual addressing scheme and segmentation as well as a 'virtually' unlimited number of erase cycles.
EcuC	0..1	Virtual module to collect ECU Configuration specific / global configuration information.
EcuM	0..1	Configuration of the EcuM (ECU State Manager) module.
Eth	0..*	Configuration of the Eth (Ethernet Driver) module.
EthIf	0..1	Configuration of the EthIf (Ethernet Interface) module.
EthSM	0..1	Configuration of the Ethernet State Manager
EthSwt	0..*	Configuration of the EthSwt (Ethernet Switch Driver) module.
EthTSyn	0..1	Configuration of the Timesync over Ethernet (EthTSyn) module with respect to global time synchronization.
EthTrcv	0..*	Configuration of Ethernet Transceiver Driver module
Fee	0..1	Configuration of the Fee (Flash EEPROM Emulation) module.
FiM	0..1	Configuration of the FiM (Function Inhibition Manager) module.
Firewall	0..1	Configuration of the Firewall module.
FlsTst	0..1	Configuration of the FlsTst module.
Fr	0..*	Configuration of the Fr (FlexRay driver) module.
FrIf	0..1	Configuration of the FrIf (FlexRay Interface) module.
FrNm	0..1	The Flexray Nm module
FrSM	0..1	Configuration of the FlexRay State Manager
FrTSyn	0..1	This represents the specific configuration variant for the TSyn on Flexray.
FrTp	0..1	Configuration of the FlexRay Transport Protocol module according to ISO 10681-2.
FrTrcv	0..*	Configuration of the FrTrcv (FlexRay Transceiver driver) module.
Gpt	0..1	Configuration of the Gpt (General Purpose Timer) module.
I2C	0..1	Configuration of the I2C (Inter-Integrated Circuit) module.
IEEE1722Tp	0..1	Configuration of the IEEE1722Tp module.
IKE	0..1	Description for the Internet Key Exchange.
ISO15118Chrg	0..1	Configuration of the ISO15118 module.
Icu	0..1	Configuration of the Icu (Input Capture Unit) module.
IdsM	0..1	Configuration of the IdsM module.
IpduM	0..1	Configuration of the IpduM (Ipdu Multiplexer) module.
J1939Dcm	0..1	The SAE J1939 Dcm module
J1939Fscp	0..1	The J1939 Functional Safety Communication Protocol covers the handling of SHM and SDM messages for E2E protected communication according to SAE J1939-76.
J1939Nm	0..1	Configuration of the J1939 Network Management module.
J1939Rm	0..1	Configuration of the J1939 Request Manager.
J1939Tp	0..1	Configuration of the J1939Tp (J1939 Transport Protocol) module.
KeyM	0..1	Configuration of the Mcu (Microcontroller Unit) module.
LSduR	0..1	Configuration of the LSduR module.
Lin	0..*	Configuration of the Lin (LIN driver) module.

▽

△

Included Modules		
Module Name	**Multiplicity**	**Description**
LinIf	0..1	Configuration of the LinIf (LIN Interface) module.
LinSM	0..1	Configuration of the Lin State Manager module.
LinTp	0..1	Configuration of the LIN Transport Protocol.
LinTrcv	0..*	Configuration of LIN Transceiver Driver module
Mcu	0..1	Configuration of the Mcu (Microcontroller Unit) module.
Mem	0..*	Configuration of the Mem driver (internal or external memory driver) module. Its multiplicity describes the actual number of Mem drivers. There will be one container for each Mem driver.
MemAcc	0..1	The MemAcc module shall only coordinate conflicting resource accesses. The access dependencies shall be configured based on MemAccSynchronizationGroupConfiguration. Note: Only relevant resource conflicts shall be coordinated to prevent any performance impact.
MemIf	0..1	Configuration of the MemIf (Memory Abstraction Interface) module.
MemMap	0..1	Configuration of the Memory Mapping module.
Mirror	0..1	Configuration of the Bus Mirroring module.
Mka	0..1	Configuration of the MACsec Key Agreement module.
Nm	0..1	The Generic Network Management Interface module
NvM	0..1	Configuration of the NvM (NvRam Manager) module.
Ocu	0..1	Configuration of Ocu (Output Compare Unit) module.
Os	0..1	Configuration of the Os (Operating System) module.
PduR	0..1	Configuration of the PduR (PDU Router) module.
Port	0..1	Configuration of the Port module.
Pwm	0..*	Configuration of Pwm (Pulse Width Modulation) module.
RamTst	0..1	Configuration of the RamTst module.
Rte	0..1	Configuration of the Rte (Runtime Environment) module.
Sd	0..1	Configuration of the Service Discovery module.
SecOC	0..1	Configuration of the SecOC (SecureOnboardCommunication) module.
SoAd	0..1	Configuration of the SoAd (Socket Adaptor) module.
SomeIpTp	0..1	Configuration of the SomeIpTp module.
Spi	0..1	Configuration of the Spi (Serial Peripheral Interface) module.
StbM	0..1	Configuration of the Synchronized Time-base Manager (StbM) module.
SwCluC	0..1	Module to collect Software Cluster Connection specific configuration information.
TcpIp	0..1	Configuration of the TcpIp (TCP/IP stack) module.
Tm	0..1	Configuration of the Time Service module.
UdpNm	0..1	Configuration of the UdpNm module.
VdpCmR	0..1	The Vehicle Data Protocol is a protocol to sample data on ECUs with low computational overhead and memory usage. It is configured using this module.
Wdg	0..*	Configuration of the Wdg (Watchdog driver) module.
WdgIf	0..1	Configuration of the WdgIf (Watchdog Interface) module.
WdgM	0..1	Configuration of the WdgM (Watchdog Manager) module.

▽

△

Included Modules		
Module Name	**Multiplicity**	**Description**
Xcp	0..1	Configuration of the XCP module
Xfrm	0..*	Configuration of the Xfrm module.

┘

3.3 Virtual Module EcuC

In the configuration of an ECU there is information which needs to be shared between multiple BSW Modules. Since it can not be defined who owns this shared information the *virtual* module EcuC has been introduced to the AUTOSAR ECU Configuration Parameter Definition.

[ECUC_EcuC_00008] Definition of EcucModuleDef EcuC ⌈

Module Name	EcuC
Description	Virtual module to collect ECU Configuration specific / global configuration information.
Post-Build Selectable Support	true
Post-Build Loadable Support	true

Included Containers		
Container Name	**Multiplicity**	**Description**
EcucBusToPartitionMappingSet	0..1	Collection of Bus to EcucPartition mappings
EcucConfigSet	0..1	This container contains the configuration parameters and sub containers of the global PduCollection.
EcucHardware	1	Hardware definition of this Ecu.
EcucPartitionCollection	0..1	Collection of Partitions defined for this ECU.
EcucPostBuildVariants	0..1	Collection of toplevel PostBuildSelectable variants. The PredefinedVariants linked inside this container will determine how many PostBuildSelectableVariants exist. If this container exist the name pattern for initialization of BSW modules will be <Mip>_Config_<PredefinedVariant.shortName>. If this container does not exist the name pattern for initialization of BSW modlues will be <Mip>_Config.
EcucUnitGroupAssignment	0..1	Collection of UnitGroup references to support the generation of ASAM MCD file.
EcucVariationResolver	0..1	Collection of PredefinedVariant elements containing definition of values for SwSystemconst which shall be applied when resolving the variability during ECU Configuration.

⌋

[ECUC_EcuC_00061] Definition of EcucParamConfContainerDef EcucConfigSet ⌈

Container Name	EcucConfigSet
Parent Container	EcuC
Upstream mapping defined	–
Description	This container contains the configuration parameters and sub containers of the global PduCollection.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

No Included Parameters

Included Containers		
Container Name	**Multiplicity**	**Description**
EcucPduCollection	0..1	Collection of all Pdu objects flowing through the Com-Stack.

⌋

3.3.1 Hardware description

In order to allow the unique description and access to hardware resources the EcucHardware has been introduced.

The EcucHardware and EcucCoreDefinition containers are always present even if the content is only required in multi-core systems. This allows an easier migration of projects to multi-core systems.

One section of the EcucHardware is concerned with the definition of computation cores and the assignment of unique EcucCoreIds to these cores. Additionally it is possible to refer to the Ecu Resource Template HwElement which represents the core in hardware.

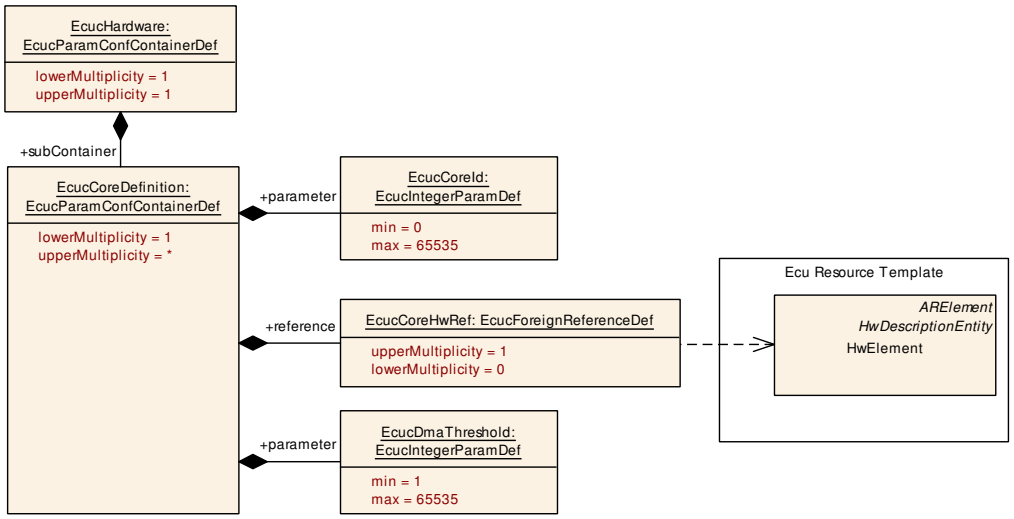

Figure 3.3: Description of ECU Hardware

[ECUC_EcuC_00056] Definition of EcucParamConfContainerDef EcucHardware
⌈

Container Name	EcucHardware
Parent Container	EcuC
Upstream mapping defined	–
Description	Hardware definition of this Ecu.
Multiplicity	1
Pre-Compile only	true
Configuration Parameters	

No Included Parameters

Included Containers		
Container Name	**Multiplicity**	**Description**
EcucCoreDefinition	1..*	Definition of one Core on this Ecu.

⌋

[ECUC_EcuC_00057] Definition of EcucParamConfContainerDef EcucCoreDefinition ⌈

Container Name	EcucCoreDefinition
Parent Container	EcucHardware
Upstream mapping defined	–
Description	Definition of one Core on this Ecu.
Multiplicity	1..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
EcucCoreId	1	[ECUC_EcuC_00058]
EcucDmaThreshold	1	[ECUC_EcuC_00094]
EcucCoreHwRef	0..1	[ECUC_EcuC_00059]

No Included Containers

⌋

[ECUC_EcuC_00058] Definition of EcucIntegerParamDef EcucCoreId ⌈

Parameter Name	EcucCoreId	
Parent Container	EcucCoreDefinition	
Upstream mapping defined	[1, CP_TPS_SystemTemplate]	
Description	ID of the core.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 65535	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_EcuC_00094] Definition of EcucIntegerParamDef EcucDmaThreshold ⌈

Parameter Name	EcucDmaThreshold
Parent Container	EcucCoreDefinition
Upstream mapping defined	–

▽

△

Description	Least number of Bytes to consider for copying asynchronously by DMA. Any number of Bytes lower than this value are copied directly and synchronously. Dependency: Only used when at least one Pdu has KeepLocalPduBuffer set to TRUE.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	1 .. 65535	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_EcuC_00059] Definition of EcucForeignReferenceDef EcucCoreHwRef ⌈

Parameter Name	EcucCoreHwRef
Parent Container	EcucCoreDefinition
Upstream mapping defined	–
Description	Optional reference to the HwElement of HwCategory ProcessingUnit that represents this Core in the ECU Resource Template.
Multiplicity	0..1
Type	Foreign reference to HW-ELEMENT
Pre-Compile only	true

⌋

3.3.2 Definition of Partitions

In order to support *memory-partitioning* and *multi-core* the notion of a *EcucPartition* has been introduced into the EcuC virtual Module.

The EcuC Module can have one `EcucPartitionCollection` which can hold an arbitrary number of `EcucPartition` elements. The *memory-partitioning* enables to create protection boundaries around groups of SWCs. The allocation of SWCs to `EcucPartition`s is possible via the `EcucPartitionSoftwareComponentInstanceRef` reference to SW Component instances. An `EcucPartition` is implemented by an OS-Application within the OS. Therefore the mapping of SWCs to partitions restricts the runnable to task mapping as shown in figure 3.4.

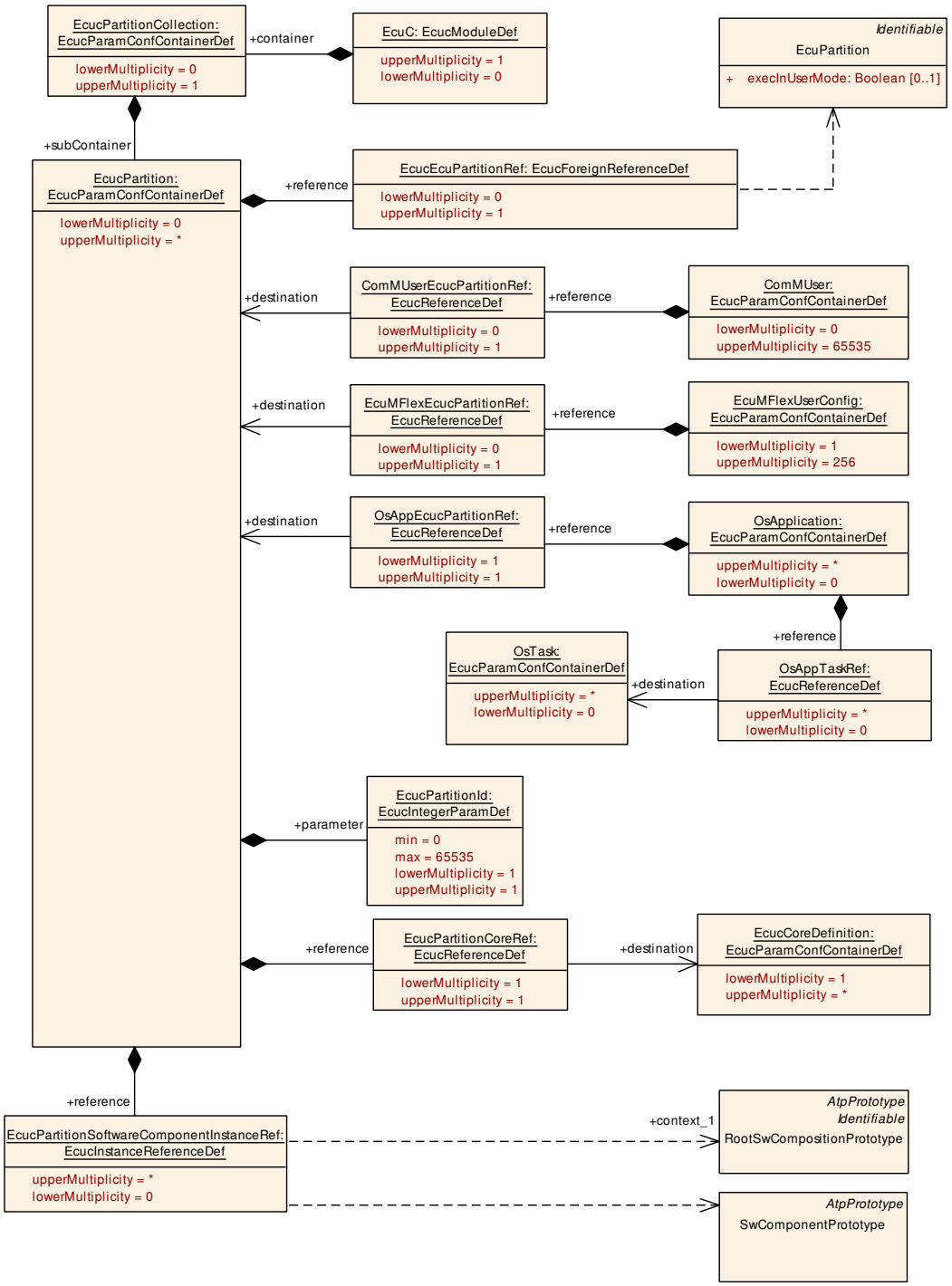

Figure 3.4: Definition of Partitions on one ECU

[ECUC_EcuC_00007] Definition of EcucParamConfContainerDef EcucPartition Collection ⌈

Container Name	EcucPartitionCollection
Parent Container	EcuC
Upstream mapping defined	–
Description	Collection of Partitions defined for this ECU.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

No Included Parameters

Included Containers		
Container Name	Multiplicity	Description
EcucPartition	0..*	Definition of one Partition on this ECU. One Partition will be implemented using one Os-Application.

⌋

[ECUC_EcuC_00005] Definition of EcucParamConfContainerDef EcucPartition ⌈

Container Name	EcucPartition
Parent Container	EcucPartitionCollection
Upstream mapping defined	[1, CP_TPS_SystemTemplate]
Description	Definition of one Partition on this ECU. One Partition will be implemented using one Os-Application.
Multiplicity	0..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
EcucPartitionId	1	[ECUC_EcuC_00085]
EcucEcuPartitionRef	0..1	[ECUC_EcuC_00083]
EcucPartitionBswModuleDistinguishedPartition	0..*	[ECUC_EcuC_00068]
EcucPartitionCoreRef	1	[ECUC_EcuC_00086]
EcucPartitionSoftwareComponentInstanceRef	0..*	[ECUC_EcuC_00036]

No Included Containers

⌋

[ECUC_EcuC_00085] Definition of EcucIntegerParamDef EcucPartitionId ⌈

Parameter Name	EcucPartitionId
Parent Container	EcucPartition
Upstream mapping defined	–
Description	ID of the partition.
Multiplicity	1
Type	EcucIntegerParamDef

▽

△

Range	0 .. 65535	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_EcuC_00083] Definition of EcucForeignReferenceDef EcucEcuPartitionRef

Status: DRAFT

⌈

Parameter Name	EcucEcuPartitionRef
Parent Container	EcucPartition
Upstream mapping defined	–
Description	Reference to the EcuPartition to define the link to the partition described in the System description. **Tags:** atp.Status=draft
Multiplicity	0..1
Type	Foreign reference to ECU-PARTITION
Pre-Compile only	true

⌋

[ECUC_EcuC_00068] Definition of EcucForeignReferenceDef EcucPartitionBswModuleDistinguishedPartition ⌈

Parameter Name	EcucPartitionBswModuleDistinguishedPartition
Parent Container	EcucPartition
Upstream mapping defined	[13, CP_TPS_BSWModuleDescriptionTemplate]
Description	This maps the abstract partition of the Bsw Module to a concrete Partition existing in the ECU.
Multiplicity	0..*
Type	Foreign reference to BSW-DISTINGUISHED-PARTITION
Pre-Compile only	true

⌋

[ECUC_EcuC_00086] Definition of EcucReferenceDef EcucPartitionCoreRef ⌈

Parameter Name	EcucPartitionCoreRef
Parent Container	EcucPartition
Upstream mapping defined	–
Description	Reference to the core definition. This reference is used to describe to which core the EcucPartition is bound.
Multiplicity	1
Type	Reference to EcucCoreDefinition
Pre-Compile only	true

⌋

[ECUC_EcuC_00036] Definition of EcucInstanceReferenceDef EcucPartitionSoftwareComponentInstanceRef ⌈

Parameter Name	EcucPartitionSoftwareComponentInstanceRef
Parent Container	EcucPartition
Upstream mapping defined	[11, CP_TPS_SoftwareComponentTemplate]
Description	References the SW Component instances from the Ecu Extract that shall be executed in this partition.
Multiplicity	0..*
Type	Instance reference to SW-COMPONENT-PROTOTYPE context: ROOT-SW-COMPOSITION-PROTOTYPE
Pre-Compile only	true

⌋

The design principle is that after the creation of a partition the software (SWC) is mapped to this partition. In the second step the BSW is configured and every member of a partition (BSW) defines a reference to the `EcucPartition` element.

One example is the Os module: The Os-Application is used to implement one Partition, therefore there shall be a reference from each Os-Application to one Partition which specifies which partition this Os-Application is implementing.

Another example is the interaction of a SWC with the ComM: A SWC running in a partition other than the BSW modules is requesting full communication at the ComM. If now the partition which the SWC is running in will be stopped due to an partition violation there is now an outstanding full communication request at the ComM which will prohibit a network to be sent to sleep. With the provided configuration means it is possible to implement counter measures for such use-cases.

The interaction between `EcucPartition` and `EcucCoreDefinition` is done via the `OsApplicationCoreRef` of `OsApplication`.

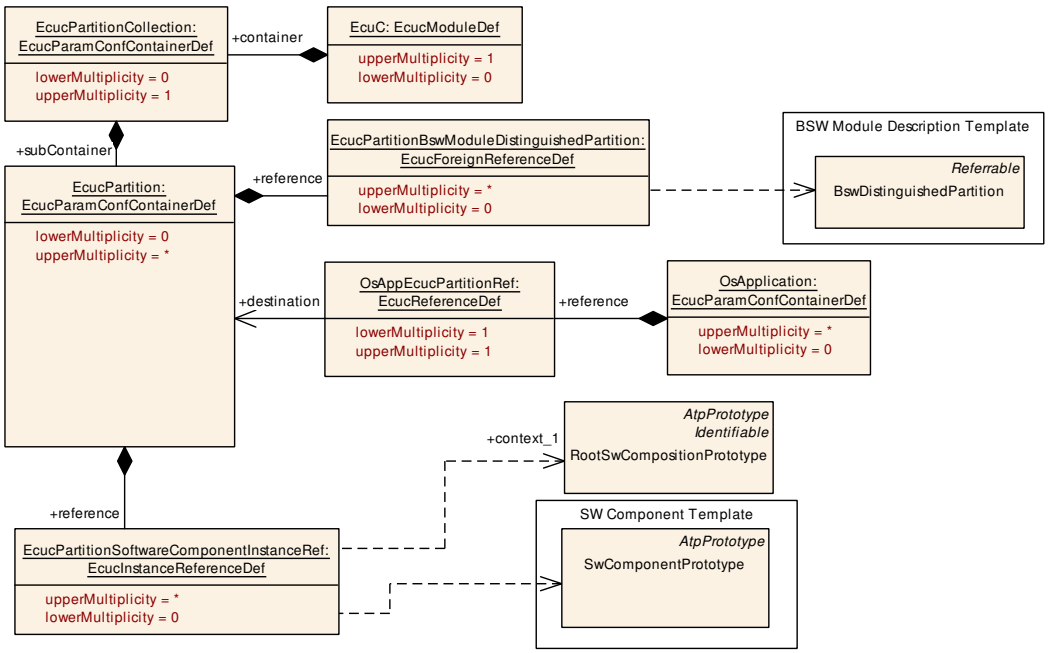

Figure 3.5: Interaction between `EcucPartition` and `EcucCoreDefinition`

3.3.3 PostBuild Variants

For each post-build variant (post-build configuration set) there exists exactly one "top-level" `PredefinedVariant` that is valid for all post-build capable BSW modules. This means that every module which supports post-build variants (previously known as post-build selectable configuration sets) will need to have configurations for every single defined `PredefinedVariant` that is referenced by `EcucPostBuildVariantRef`.

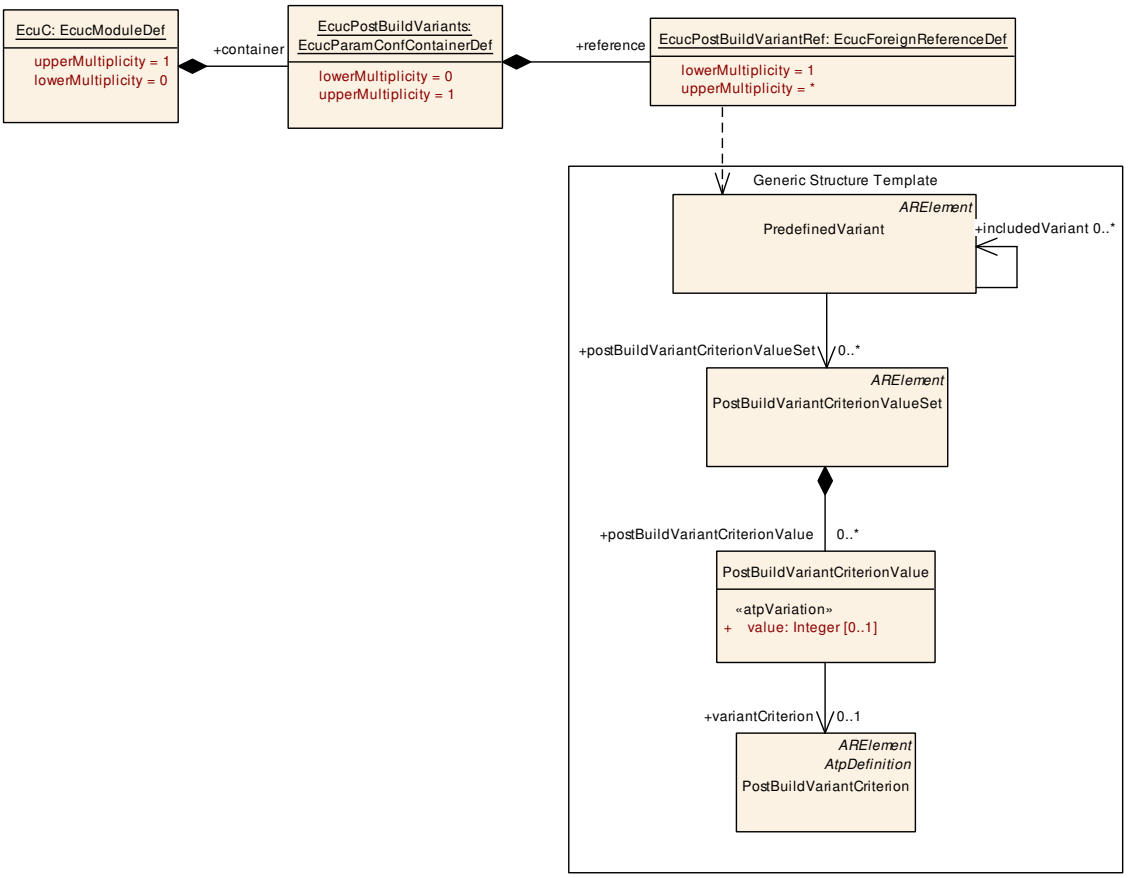

Figure 3.6: Collection of toplevel PostBuildSelectable variants

[ECUC_EcuC_00070] Definition of EcucParamConfContainerDef EcucPostBuild Variants ⌈

Container Name	EcucPostBuildVariants
Parent Container	EcuC
Upstream mapping defined	–
Description	Collection of toplevel PostBuildSelectable variants. The PredefinedVariants linked inside this container will determine how many PostBuildSelectableVariants exist. If this container exist the name pattern for initialization of BSW modules will be <Mip>_ Config_<PredefinedVariant.shortName>. If this container does not exist the name pattern for initialization of BSW modlues will be <Mip>_Config.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
EcucPostBuildVariantRef	1..*	[ECUC_EcuC_00071]

No Included Containers

⌋

[ECUC_EcuC_00071] Definition of EcucForeignReferenceDef EcucPostBuildVariantRef ⌈

Parameter Name	EcucPostBuildVariantRef
Parent Container	EcucPostBuildVariants
Upstream mapping defined	[14, CP_TPS_ECUConfiguration]
Description	Reference to a PredefinedVariant that defines one toplevel postBuild configuration set (covering all post-build capable BSW modules). PredefinedVariants that are referenced here shall contain only PostBuildVariantCriterionValueSets.
Multiplicity	1..*
Type	Foreign reference to PREDEFINED-VARIANT
Pre-Compile only	true

⌋

[constr_3307] ShortNames of `PredefinedVariant`s referenced by `EcucPostBuildVariantRef`s

Imposition time: CP: `IT_EcucVal`

⌈All `PredefinedVariant`s that are referenced by `EcucPostBuildVariantRef`s shall have different `shortName`s.⌋

`PredefinedVariant`s may exist in different packages and thus have the same `shortName`. The generation of symbols in `EcucPostBuildVariants` requires these `shortName`s to be different.

3.3.4 Variation Resolver Description

In order to support the variant handling approach (see Generic Structure Template [4]) the already given values of system constants are specified in using the collection `SwSystemconstantValueSet`. In the `EcuC` the applicable `SwSystemconstantValueSet` elements are referenced indirectly via the `PredefinedVariant` collection.

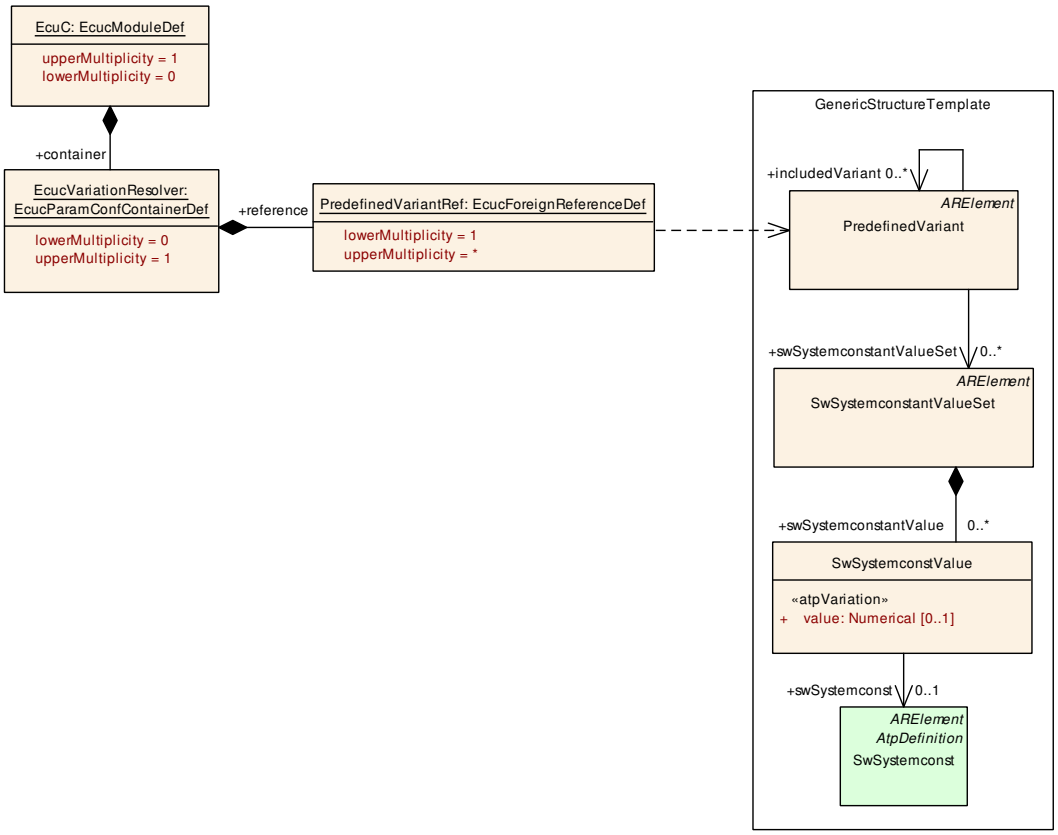

Figure 3.7: Description of Variation Resolver

[ECUC_EcuC_00009] Definition of EcucParamConfContainerDef EcucVariation Resolver ⌈

Container Name	EcucVariationResolver
Parent Container	EcuC
Upstream mapping defined	–
Description	Collection of PredefinedVariant elements containing definition of values for Sw Systemconst which shall be applied when resolving the variability during ECU Configuration.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
PredefinedVariantRef	1..*	[ECUC_EcuC_00010]

No Included Containers

⌋

[ECUC_EcuC_00010] Definition of EcucForeignReferenceDef PredefinedVariantRef ⌈

Parameter Name	PredefinedVariantRef
Parent Container	EcucVariationResolver
Upstream mapping defined	[14, CP_TPS_ECUConfiguration]
Description	Reference to a PredefinedVariant
Multiplicity	1..*
Type	Foreign reference to PREDEFINED-VARIANT
Pre-Compile only	true

⌋

3.3.5 UnitGroup Assignment

To support the generation of ASAM MCD files `UnitGroup`s may be selected in the EcuC that are relevant for the MCD system. Please note that the `EcucUnitGroup-Assignment` can be used to control the generation of the A2L file in a way that the units used for calculation are replaced by application domain specific units.

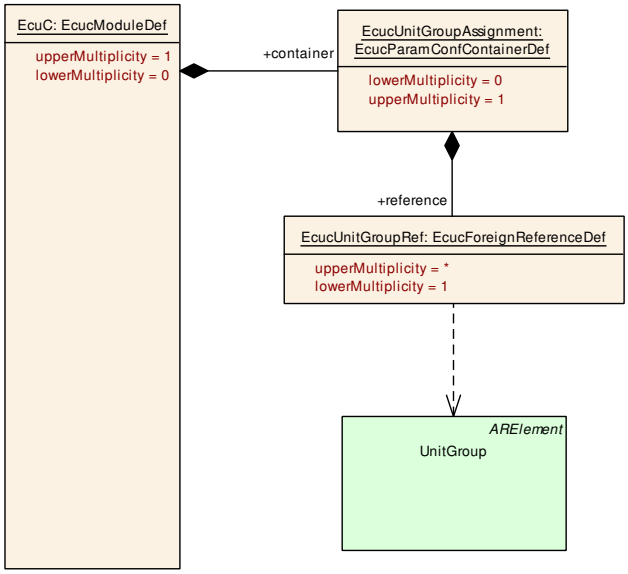

Figure 3.8: Assignment of UnitGroups

[ECUC_EcuC_00063] Definition of EcucParamConfContainerDef EcucUnitGroupAssignment ⌈

Container Name	EcucUnitGroupAssignment
Parent Container	EcuC
Upstream mapping defined	–
Description	Collection of UnitGroup references to support the generation of ASAM MCD file.
Multiplicity	0..1

▽

△

Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
EcucUnitGroupRef	1..*	[ECUC_EcuC_00062]

No Included Containers

⌋

[ECUC_EcuC_00062] Definition of EcucForeignReferenceDef EcucUnitGroupRef
⌈

Parameter Name	EcucUnitGroupRef
Parent Container	EcucUnitGroupAssignment
Upstream mapping defined	[11, CP_TPS_SoftwareComponentTemplate], [13, CP_TPS_BSWModuleDescription Template]
Description	Optional reference to the UnitGroup to support the generation of ASAM MCD file. These UnitGroups are selecting a set of units for a specific country.
Multiplicity	1..*
Type	Foreign reference to UNIT-GROUP
Pre-Compile only	true

⌋

3.3.6 Bus to Partition Mapping

The bus to partition mapping allows to map a communication bus to an `EcucPartition`, by creating an `EcucBusToPartitionMapping`, whose `EcucBusToPartitionMappingBusRef` refers to the `ComMChannel` representing that bus. This can then be used to derive further configuration for an `EcuInstance`. For details, see [15] Chapter 2.7.5.2 *"Using EcucBusToPartitionMapping"*.

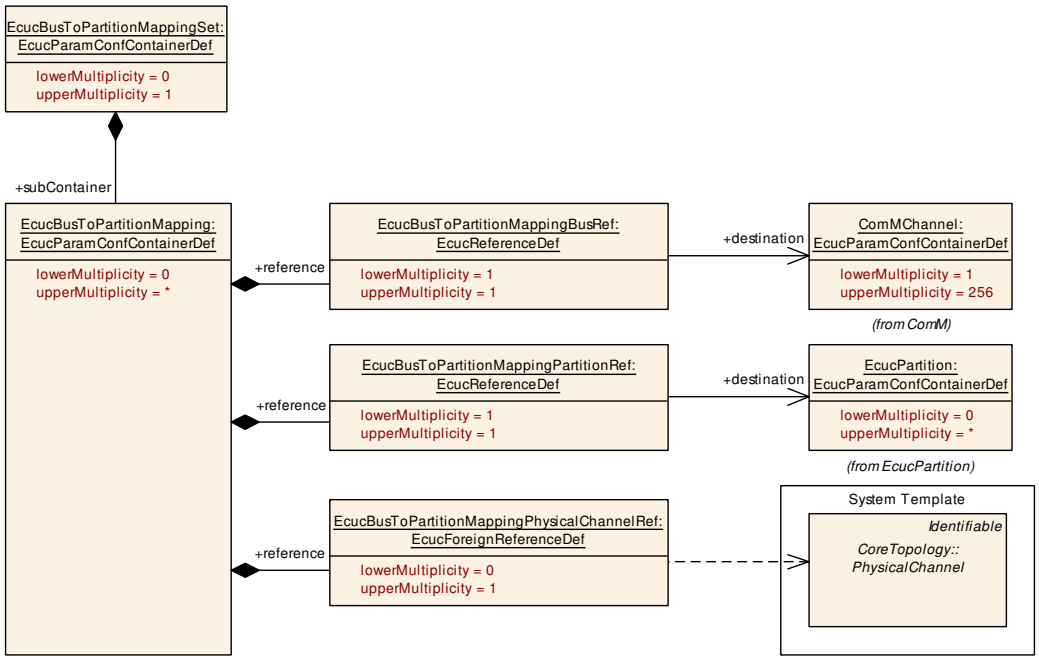

Figure 3.9: Ecuc Bus to Partition Mapping

[ECUC_EcuC_00089] Definition of EcucParamConfContainerDef EcucBusToPartitionMappingSet ⌈

Container Name	EcucBusToPartitionMappingSet
Parent Container	EcuC
Upstream mapping defined	–
Description	Collection of Bus to EcucPartition mappings
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

No Included Parameters

Included Containers		
Container Name	**Multiplicity**	**Description**
EcucBusToPartitionMapping	0..*	Map a communication bus to an EcucPartition

⌋

[ECUC_EcuC_00090] Definition of EcucParamConfContainerDef EcucBusToPartitionMapping ⌈

Container Name	EcucBusToPartitionMapping
Parent Container	EcucBusToPartitionMappingSet
Upstream mapping defined	–
Description	Map a communication bus to an EcucPartition

▽

△

Multiplicity	0..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
EcucBusToPartitionMappingBusRef	1	[ECUC_EcuC_00091]
EcucBusToPartitionMappingPartitionRef	1	[ECUC_EcuC_00092]
EcucBusToPartitionMappingPhysicalChannelRef	0..1	[ECUC_EcuC_00093]

No Included Containers

⌋

[ECUC_EcuC_00091] Definition of EcucReferenceDef EcucBusToPartitionMappingBusRef ⌈

Parameter Name	EcucBusToPartitionMappingBusRef
Parent Container	EcucBusToPartitionMapping
Upstream mapping defined	–
Description	Reference to ComMChannel that represents this communication bus, as it is part of the control path for that bus. Note that the EcucBusToPartitionMapping refers to the partition that the data path (e.g. bus driver, interface, etc.) is executed in. The control path, and specifically the ComM BSW module, does not have to be executed in the same partition as the data path. So it is possible that the EcucBusToPartitionMapping refers to a partition that is a different partition than the one which ComM runs in. Example: the ComM BSW module is executed in partition_3, while the EcucBusToPartitionMapping refers to partition_8.
Multiplicity	1
Type	Reference to ComMChannel
Pre-Compile only	true

⌋

[ECUC_EcuC_00092] Definition of EcucReferenceDef EcucBusToPartitionMappingPartitionRef ⌈

Parameter Name	EcucBusToPartitionMappingPartitionRef
Parent Container	EcucBusToPartitionMapping
Upstream mapping defined	–
Description	Reference to the EcucPartition
Multiplicity	1
Type	Reference to EcucPartition
Pre-Compile only	true

⌋

[ECUC_EcuC_00093] Definition of EcucForeignReferenceDef EcucBusToPartitionMappingPhysicalChannelRef ⌈

Parameter Name	EcucBusToPartitionMappingPhysicalChannelRef
Parent Container	EcucBusToPartitionMapping
Upstream mapping defined	–
Description	Optional reference to a PhysicalChannel. On the system level, this allows finding the relevant Pdus via PhysicalChannel.pduTriggering. The PduTriggerings can then be cross referenced to Ecuc Pdus.
Multiplicity	0..1
Type	Foreign reference to PHYSICAL-CHANNEL
Pre-Compile only	true

⌋

3.3.7 Definition of Pdus

In order to support the synchronization of Handle IDs (see section 3.4.1) two modules need to be able to refer to the same Pdu object[2]. Therefore a generic `Pdu` container has been defined which does not belong to any module but is defined in the `EcuC` module.

Since the Pdu flowing through the COM-Stack does not belong to an individual module, the "virtual" module `EcuC` has been introduced in the ECU Configuration. This module is used to collect configuration information not associated with any specific standardized module.

The `EcucPduCollection` may contain several "global" `Pdu` objects as shown in figure 3.10. Each `Pdu` may either represent a `FrameTriggering` (for `Pdu`s not going through the Pdu Router: `UserDefinedPdu`s, `NmPdu`s and `NPdu`s) or `PduTriggering` (for all other `Pdu`s) belonging to the specific `ECU` from the AUTOSAR `System Description`[1] (`ECU Extract`). Therefore there is an optional reference to either `FrameTriggering` (`SysTPduToFrameTriggeringRef`) or `PduTriggering` (`SysTPduToPduTriggeringRef`) element in the `System Template`. Either `SysTPduToFrameTriggeringRef` or `SysTPduToPduTriggeringRef` shall be used.

[2]For the aspect of the configuration it does not matter what kind of Pdu it is, i.e. I-PDU, L-PDU or N-PDU.

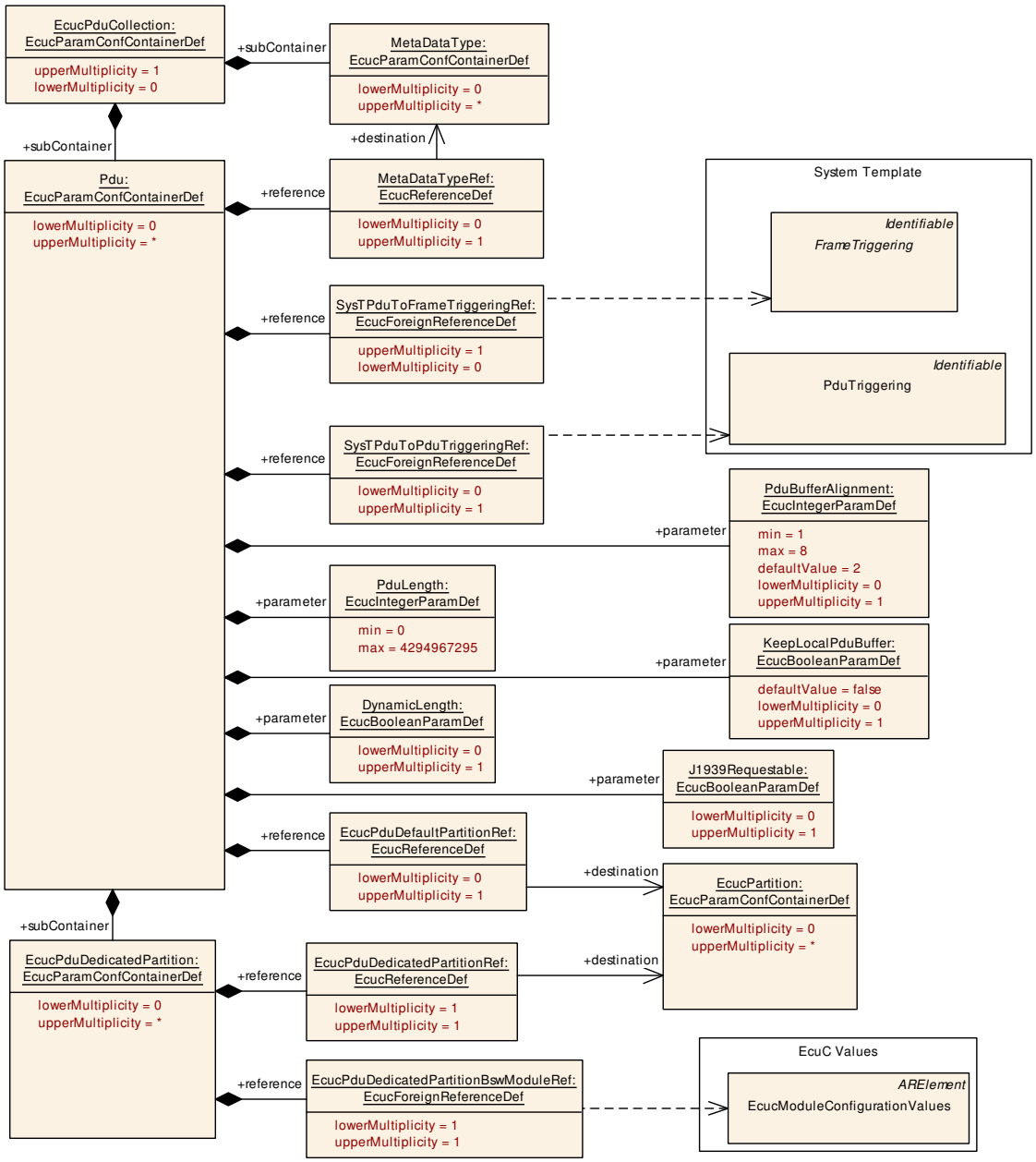

Figure 3.10: Generic Pdu Container

[ECUC_EcuC_00002] Definition of EcucParamConfContainerDef EcucPduCollection ⌈

Container Name	EcucPduCollection
Parent Container	EcucConfigSet
Upstream mapping defined	–
Description	Collection of all Pdu objects flowing through the Com-Stack.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

No Included Parameters

Included Containers		
Container Name	**Multiplicity**	**Description**
MetaDataType	0..*	Meta data serves to transport information through the AUTOSAR layers. It is transported by the PduInfoType structure via a separate pointer to a byte array alongside the length of and a pointer to the payload of the PDU. This container defines the content of the meta data.
Pdu	0..*	One Pdu flowing through the COM-Stack. This Pdu is used by all Com-Stack modules to agree on referencing the same Pdu.

⌋

[ECUC_EcuC_00001] Definition of EcucParamConfContainerDef Pdu ⌈

Container Name	Pdu	
Parent Container	EcucPduCollection	
Upstream mapping defined	–	
Description	One Pdu flowing through the COM-Stack. This Pdu is used by all Com-Stack modules to agree on referencing the same Pdu.	
Multiplicity	0..*	
Post-Build Multiplicity	Loadable	Selectable
Configuration Parameters		

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DynamicLength	0..1	[ECUC_EcuC_00078]
J1939Requestable	0..1	[ECUC_EcuC_00072]
KeepLocalPduBuffer	0..1	[ECUC_EcuC_00087]
PduBufferAlignment	0..1	[ECUC_EcuC_00088]
PduLength	1	[ECUC_EcuC_00003]
EcucPduDefaultPartitionRef	0..1	[ECUC_EcuC_00082]
MetaDataTypeRef	0..1	[ECUC_EcuC_00077]
SysTPduToFrameTriggeringRef	0..1	[ECUC_EcuC_00052]
SysTPduToPduTriggeringRef	0..1	[ECUC_EcuC_00054]

Included Containers		
Container Name	**Multiplicity**	**Description**
EcucPduDedicatedPartition	0..*	Module specific container for Pdu to partition assignment.

⌋

[ECUC_EcuC_00078] Definition of EcucBooleanParamDef DynamicLength ⌈

Parameter Name	DynamicLength	
Parent Container	Pdu	
Upstream mapping defined	[1, CP_TPS_SystemTemplate]	
Description	This parameter defines whether the Pdu has dynamic length (true) or not (false). Please note that the usage of this attribute is restricted by [constr_3448].	
Multiplicity	0..1	
Type	EcucBooleanParamDef	
Default value	–	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

⌋

[ECUC_EcuC_00072] Definition of EcucBooleanParamDef J1939Requestable ⌈

Parameter Name	J1939Requestable
Parent Container	Pdu
Upstream mapping defined	[1, CP_TPS_SystemTemplate]

▽

△

Description	Pdu can be triggered by the J1939 request message.	
Multiplicity	0..1	
Type	EcucBooleanParamDef	
Default value	–	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

⌋

[ECUC_EcuC_00087] Definition of EcucBooleanParamDef KeepLocalPduBuffer

Status: DRAFT

⌈

Parameter Name	KeepLocalPduBuffer
Parent Container	Pdu
Upstream mapping defined	–
Description	This parameter defines whether a module that handles the PDU would keep a temporary local buffer • until a confirmation or release-rx-buffer function call arrives (KeepLocalPduBuffer = TRUE) or • if a temporary local buffer is released after transmission or rx indication return (Keep LocalPduBuffer = FALSE). **Tags:** atp.Status=draft
Multiplicity	0..1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

⌋

[ECUC_EcuC_00088] Definition of EcucIntegerParamDef PduBufferAlignment

Status: DRAFT

⌈

Parameter Name	PduBufferAlignment	
Parent Container	Pdu	
Upstream mapping defined	–	
Description	This parameter defines the byte alignment of temporary local buffers that is required by the hardware. Using this parameter, a configuration can ensure that an upper layer module is aware of the alignment requirements of a driver. Unit: Byte **Tags:** atp.Status=draft Dependency: This parameter shall only be configured if KeepLocalPduBuffer is True.	
Multiplicity	0..1	
Type	EcucIntegerParamDef	
Range	1 .. 8	
Default value	2	

▽

△

Pre-Compile only	true

⌋

[ECUC_EcuC_00003] Definition of EcucIntegerParamDef PduLength ⌈

Parameter Name	PduLength	
Parent Container	Pdu	
Upstream mapping defined	[1, CP_TPS_SystemTemplate]	
Description	Length of the Pdu in bytes. It should be noted that in former AUTOSAR releases (Rel 2.1, Rel 3.0, Rel 3.1, Rel 4.0 Rev. 1) this parameter was defined in bits.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 4294967295	
Default value	–	
Post-Build Value	Loadable	Selectable

⌋

[ECUC_EcuC_00082] Definition of EcucReferenceDef EcucPduDefaultPartitionRef ⌈

Parameter Name	EcucPduDefaultPartitionRef
Parent Container	Pdu
Upstream mapping defined	–
Description	Reference to EcucPartition, where the according Pdu is assigned to.
Multiplicity	0..1
Type	Reference to EcucPartition
Pre-Compile only	true

⌋

[ECUC_EcuC_00077] Definition of EcucReferenceDef MetaDataTypeRef ⌈

Parameter Name	MetaDataTypeRef
Parent Container	Pdu
Upstream mapping defined	[11, CP_TPS_SoftwareComponentTemplate], [1, CP_TPS_SystemTemplate]
Description	Reference to meta data that is transported in the Pdu through the AUTOSAR layers.
Multiplicity	0..1
Type	Reference to MetaDataType
Pre-Compile only	true

⌋

[ECUC_EcuC_00052] Definition of EcucForeignReferenceDef SysTPduToFrameTriggeringRef ⌈

Parameter Name	SysTPduToFrameTriggeringRef	
Parent Container	Pdu	
Upstream mapping defined	[1, CP_TPS_SystemTemplate]	
Description	Reference to the FrameTriggering from the SystemTemplate which this Pdu belongs to. SysTPduToFrameTriggeringRef shall be used for UserDefinedPdus, NmPdus and NPdus which are not going through the Pdu Router. This reference shall not be used if SysTPduToPduTriggeringRef exists. Dependency: SysTPduToFrameTriggeringRef shall be used for UserDefinedPdus, Nm Pdus and NPdus which are not going through the Pdu Router. This reference shall not be used if SysTPduToPduTriggeringRef exists.	
Multiplicity	0..1	
Type	Foreign reference to FRAME-TRIGGERING	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

⌋

[ECUC_EcuC_00054] Definition of EcucForeignReferenceDef SysTPduToPduTriggeringRef ⌈

Parameter Name	SysTPduToPduTriggeringRef	
Parent Container	Pdu	
Upstream mapping defined	[1, CP_TPS_SystemTemplate]	
Description	Reference to the PduTriggering from the SystemTemplate which this Pdu represents. SysTPduToPduTriggeringRef shall be used for all Pdus except UserDefinedPdus, Nm Pdus and NPdus which are not going through the Pdu Router. For these Pdus, Sys TPduToFrameTriggeringRef shall be used. Dependency: SysTPduToPduTriggeringRef shall be used for all Pdus except User DefinedPdus, NmPdus and NPdus which are not going through the Pdu Router. This reference shall not be used if SysTPduToFrameTriggeringRef exists.	
Multiplicity	0..1	
Type	Foreign reference to PDU-TRIGGERING	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

⌋

[ECUC_EcuC_00079] Definition of EcucParamConfContainerDef EcucPduDedicatedPartition ⌈

Container Name	EcucPduDedicatedPartition	
Parent Container	Pdu	
Upstream mapping defined	–	
Description	Module specific container for Pdu to partition assignment.	
Multiplicity	0..*	
Post-Build Multiplicity	-	Selectable
Configuration Parameters		

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
EcucPduDedicatedPartitionBswModuleRef	1	[ECUC_EcuC_00080]
EcucPduDedicatedPartitionRef	1	[ECUC_EcuC_00081]

No Included Containers

⌋

[ECUC_EcuC_00080] Definition of EcucForeignReferenceDef EcucPduDedicatedPartitionBswModuleRef ⌈

Parameter Name	EcucPduDedicatedPartitionBswModuleRef
Parent Container	EcucPduDedicatedPartition
Upstream mapping defined	–
Description	Reference to BSW module, for which the according dedicated Pdu assignment is valid.
Multiplicity	1
Type	Foreign reference to ECUC-MODULE-CONFIGURATION-VALUES
Pre-Compile only	true

⌋

[ECUC_EcuC_00081] Definition of EcucReferenceDef EcucPduDedicatedPartitionRef ⌈

Parameter Name	EcucPduDedicatedPartitionRef
Parent Container	EcucPduDedicatedPartition
Upstream mapping defined	–
Description	Module specific reference to EcucPartition, where the according Pdu is assigned to. The dedicated partition reference shall overrule the default partition reference for the respective module.
Multiplicity	1
Type	Reference to EcucPartition
Pre-Compile only	true

⌋

[TPS_ECUC_06030] Invalid `PduLength` parameter value configuration ⌈Configuring the `PduLength` larger than the underlying layer supports results in an invalid configuration.⌋

3.3.8 Pdu Meta-Data

Meta-Data of `Pdu`s is supported by a large number of modules of the AUTOSAR communication stack. The Meta-Data transports information through the layers, that is in general abstracted by the layered architecture. The content of the Meta-Data is defined by `MetaDataType` and the relation to a `Pdu` is created by the `MetaDataTypeRef`.

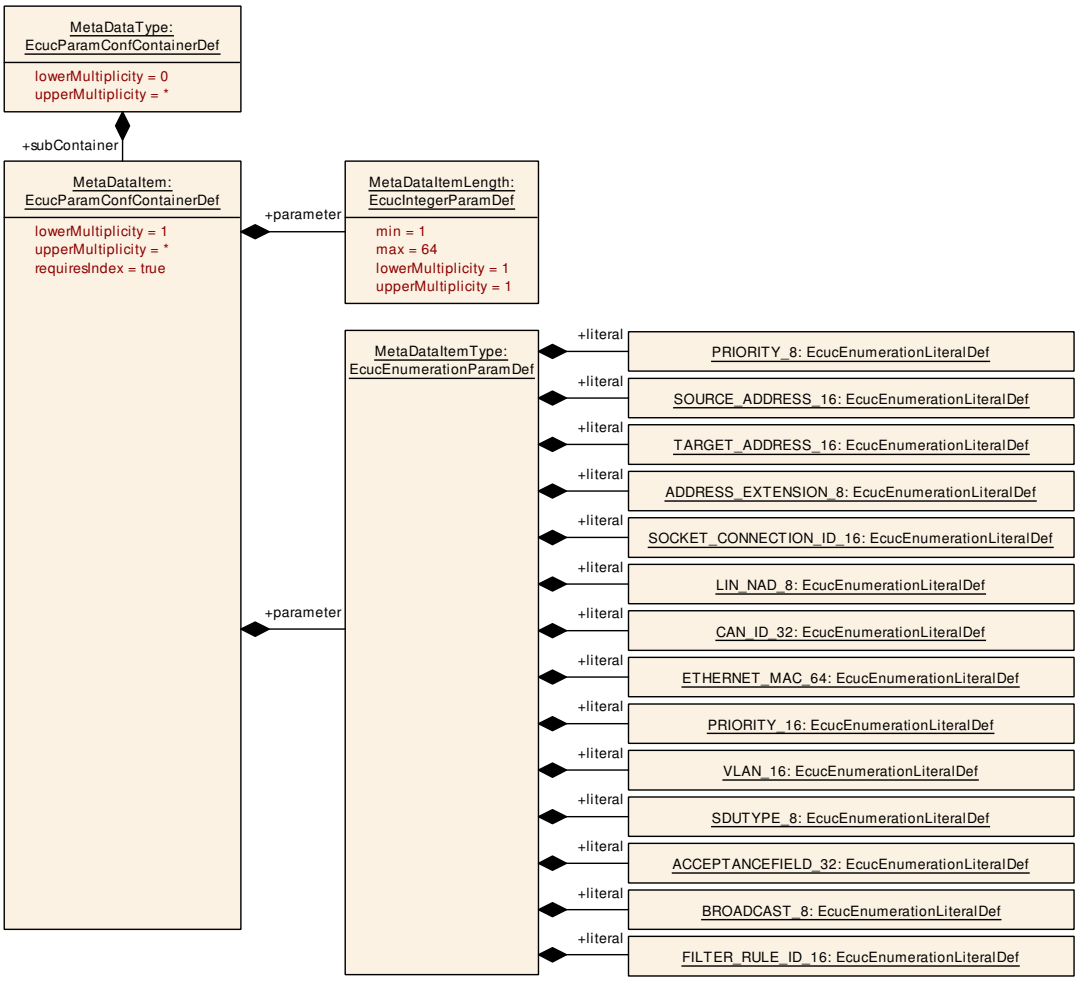

Figure 3.11: Pdu Meta-Data

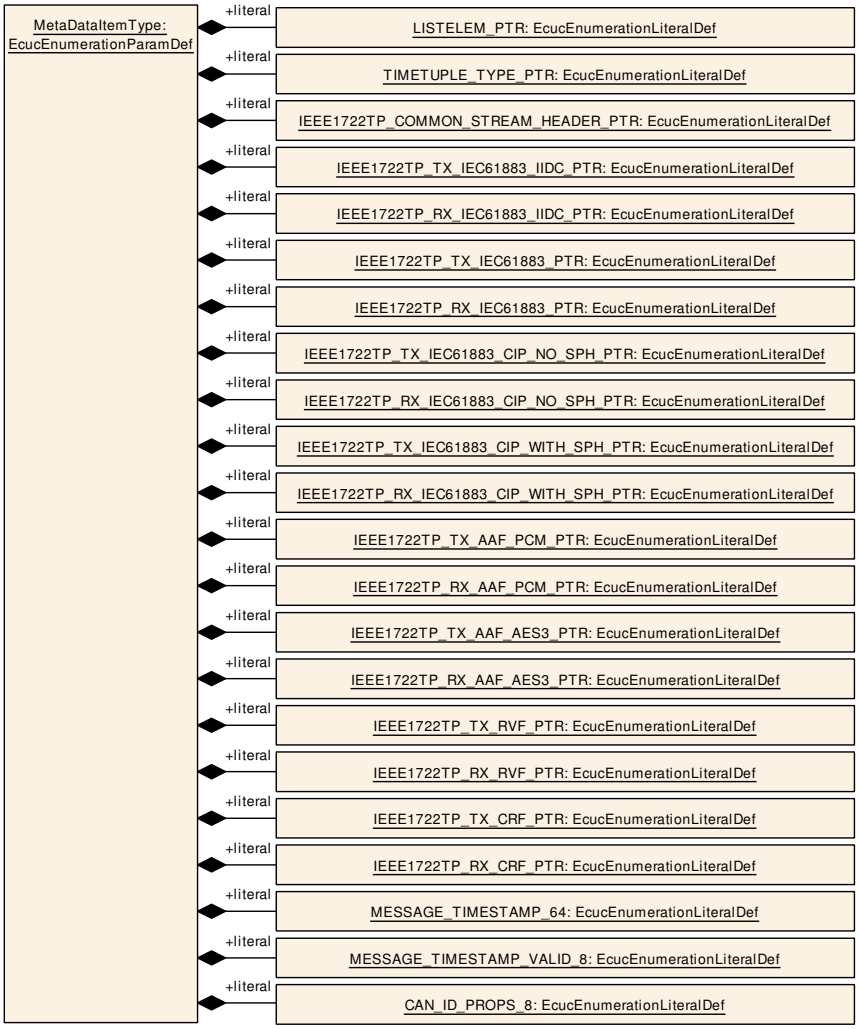

Figure 3.12: Pdu Meta-Data used for IEEE1722Tp

[ECUC_EcuC_00073] Definition of EcucParamConfContainerDef MetaDataType ⌈

Container Name	MetaDataType
Parent Container	EcucPduCollection
Upstream mapping defined	–
Description	Meta data serves to transport information through the AUTOSAR layers. It is transported by the PduInfoType structure via a separate pointer to a byte array alongside the length of and a pointer to the payload of the PDU. This container defines the content of the meta data.
Multiplicity	0..*
Pre-Compile only	true
Configuration Parameters	

No Included Parameters

Included Containers		
Container Name	**Multiplicity**	**Description**
MetaDataItem	1..*	The content of meta data in a Pdu consists of an ordered list of meta data items. This container represents a meta data item that is contained in meta data of a Pdu.

⌋

[ECUC_EcuC_00074] Definition of EcucParamConfContainerDef MetaDataItem ⌈

Container Name	MetaDataItem
Parent Container	MetaDataType
Upstream mapping defined	[11, CP_TPS_SoftwareComponentTemplate]
Description	The content of meta data in a Pdu consists of an ordered list of meta data items. This container represents a meta data item that is contained in meta data of a Pdu. **Attributes:** requiresIndex=true
Multiplicity	1..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
MetaDataItemLength	1	[ECUC_EcuC_00075]
MetaDataItemType	1	[ECUC_EcuC_00076]

No Included Containers

⌋

[ECUC_EcuC_00075] Definition of EcucIntegerParamDef MetaDataItemLength ⌈

Parameter Name	MetaDataItemLength	
Parent Container	MetaDataItem	
Upstream mapping defined	[11, CP_TPS_SoftwareComponentTemplate]	
Description	This parameter defines the length of a meta data item in bytes.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	1 .. 64	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_EcuC_00076] Definition of EcucEnumerationParamDef MetaDataItemType ⌈

Parameter Name	MetaDataItemType	
Parent Container	MetaDataItem	
Upstream mapping defined	[1, CP_TPS_SystemTemplate]	
Description	This parameter defines the type of a meta data item.	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	ACCEPTANCEFIELD_32	Acceptance field of CAN XL. Size: 32 bits.

▽

△

	ADDRESS_EXTENSION_8	Address extension field (N_AE) of the mixed addressing modes with 11bit and 29bit CAN ID of ISO 15765-2. Size: 8 bits.
	BROADCAST_8	Boolean information about whether the PDU is a broadcast PDU. **Tags:** atp.Status=draft
	CAN_ID_32	CAN ID according to ISO 11898-2, either 29 bits or 11 bits. Encoding according to Can_IdType. Size: 32 bits.
	CAN_ID_PROPS_8	Encode CAN specific information as bit field: • bit 0 == rtr (remote transmission request) • bit 1 == brs (bit rate switch) • bit 2 == esi (error state indicator) • bit 3-7 == reserved Please note: eff (extended frame format) and fdf (CAN Flexible Data-rate (FD) format) are encoded within the CAN_ID_32 at the two most significant bits. **Tags:** atp.Status=draft
	ETHERNET_MAC_64	For 48bit MAC addresses the upper 16 bit are not used by the producer and ignored by the consumer.
	FILTER_RULE_ID_16	ID of the filter rule that was applied to an Ethernet frame by a switch. **Tags:** atp.Status=draft
	IEEE1722TP_COMMON_STREAM_HEADER_PTR	Represents the runtime values for a common stream header field, provided as pointer to an IEEE1722Tp_CommonStreamHeaderType. **Tags:** atp.Status=draft
	IEEE1722TP_RX_AAF_AES3_PTR	Represents the runtime values of an AAF stream with PCM encapsulation, provided as pointer to an IEEE1722Tp_RxAafAes3Type. Used for reception, which is forwarded to an IEEE1722 data stream consumer. **Tags:** atp.Status=draft
	IEEE1722TP_RX_AAF_PCM_PTR	Represents the runtime values of an AAF stream with PCM encapsulation, provided as pointer to an IEEE1722Tp_RxAafPcmType. Used for reception, which is forwarded to an IEEE1722 data stream consumer. **Tags:** atp.Status=draft
	IEEE1722TP_RX_CRF_PTR	Represents the runtime values of a CRF stream, provided as pointer to an IEEE1722Tp_RxCrf Type. Used for reception, which is forwarded to an IEEE1722 data stream consumer. **Tags:** atp.Status=draft
	IEEE1722TP_RX_IEC61883_CIP_NO_SPH_PTR	Represents the runtime values of an IEC61883 stream with SPH not set, provided as pointer to an IEEE1722Tp_RxIec61883CipNoSphType. Used for reception, which is forwarded to an IEEE1722 data stream consumer. **Tags:** atp.Status=draft

▽

△

	IEEE1722TP_RX_IEC61883_CIP_WITH_SPH_PTR	Represents the runtime values of an IEC61883 stream with SPH set, provided as pointer to an IEEE1722Tp_RxIec61883CipWithSphType. Used for reception, which is forwarded to an IEEE1722 data stream consumer. **Tags:** atp.Status=draft
	IEEE1722TP_RX_IEC61883_IIDC_PTR	Represents the runtime values of an IEC61883/IIDC stream, provided as pointer to an IEEE1722Tp_RxIec61883IidcType. Used for reception, which is forwarded to an IEEE1722 data stream consumer. **Tags:** atp.Status=draft
	IEEE1722TP_RX_IEC61883_PTR	Represents the runtime values of an IEC61883 stream, provided as pointer to an IEEE1722Tp_RxIec61883Type. Used for reception, which is forwarded to an IEEE1722 data stream consumer. **Tags:** atp.Status=draft
	IEEE1722TP_RX_RVF_PTR	Represents the runtime values of an RVF stream, provided as pointer to an IEEE1722Tp_RxRvfType. Used for reception, which is forwarded to an IEEE1722 data stream consumer. **Tags:** atp.Status=draft
	IEEE1722TP_TX_AAF_AES3_PTR	Represents the runtime values for an AAF stream with AES3 encapsulation, provided as pointer to an IEEE1722Tp_TxAafAes3Type. Used for transmission by an IEEE1722 data stream provider. **Tags:** atp.Status=draft
	IEEE1722TP_TX_AAF_PCM_PTR	Represents the runtime values for an AAF stream with PCM encapsulation, provided as pointer to an IEEE1722Tp_TxAafPcmType. Used for transmission by an IEEE1722 data stream provider. **Tags:** atp.Status=draft
	IEEE1722TP_TX_CRF_PTR	Represents the runtime values for a CRF stream, provided as pointer to an IEEE1722Tp_TxCrfType. Used for transmission by an IEEE1722 data stream producer. **Tags:** atp.Status=draft
	IEEE1722TP_TX_IEC61883_CIP_NO_SPH_PTR	Represents the runtime values for an IEC61883 stream with SPH not set, provided as pointer to an IEEE1722Tp_TxIec61883CipNoSphType. Used for transmission by an IEEE1722 data stream provider. **Tags:** atp.Status=draft
	IEEE1722TP_TX_IEC61883_CIP_WITH_SPH_PTR	Represents the runtime values for an IEC61883 stream with SPH set, provided as pointer to an IEEE1722Tp_TxIec61883CipWithSphType. Used for transmission by an IEEE1722 data stream provider. **Tags:** atp.Status=draft

▽

△

	IEEE1722TP_TX_IEC61883_IIDC_PTR	Represents the runtime values for an IEC61883/IIDC, provided as pointer to an IEEE1722Tp_Tx Iec61883IidcType. Used for transmission by an IEEE1722 data stream provider. **Tags:** atp.Status=draft
	IEEE1722TP_TX_IEC61883_PTR	Represents the runtime values for an IEC61883 stream, provided as pointer to an IEEE1722Tp_ TxIec61883Type. Used for transmission by an IEEE1722 data stream provider. **Tags:** atp.Status=draft
	IEEE1722TP_TX_RVF_PTR	Represents the runtime values for an RVF stream, provided as pointer to an IEEE1722Tp_ TxRvfType. Used for transmission by an IEEE1722 data stream producer. **Tags:** atp.Status=draft
	LIN_NAD_8	LIN node address as used in the LIN transport protocol. Size: 8 bits.
	LISTELEM_PTR	Pointer to ListElemStructType provided by Com StackTypes. Used to produce a single linked list to support for example hardware supported data transfer within the communication stack. **Tags:** atp.Status=draft
	MESSAGE_TIMESTAMP_64	Represents the message creation time in nanoseconds. E.g. Used to convey the timestamp of a received IEEE1722 ACF-message to an IEEE1722 application. **Tags:** atp.Status=draft
	MESSAGE_TIMESTAMP_VALID_8	Represents a bit to indicate whether the MESSAGE_TIMESTAMP_64 contains valid data. E.g. Used to convey validity bit of an received IEEE1722 ACF-message to an IEEE1722 application. **Tags:** atp.Status=draft
	PRIORITY_16	CAN XL Priority ID. Size: 16 bits.
	PRIORITY_8	Priority field of SAE J1939 IDs, or Ethernet QoS parameter. Size: 8 bits.
	SDUTYPE_8	SDU type of CAN XL. Size: 8 bits.
	SOCKET_CONNECTION_ID_16	SoAd socket connection ID. Size: 16 bits.
	SOURCE_ADDRESS_16	Source address of CanTp, FrTp, or DoIP transport protocol messages, or of SAE J1939 messages. Size: 16 bits.
	TARGET_ADDRESS_16	Target address of CanTp, FrTp, or DoIP transport protocol messages, or destination address of SAE J1939 messages. Size: 16 bits.
	TIMETUPLE_TYPE_PTR	Represents a pointer to a TimeTupleType provided by ComStackTypes. Used to forward ingress and egress time stamps to the upper layer. **Tags:** atp.Status=draft
	VLAN_16	VLAN ID of Ethernet or VCID of CAN XL. Size: 16 bits.
Pre-Compile only	true	

┘

[constr_5059] Ordering of MetaDataItems of a MetaDataType

Imposition time: CP: IT_EcucVal

⌈The MetaDataItems of a MetaDataType shall be ordered according to their MetaDataItemLength. MetaDataItems with greater MetaDataItemLength going first.⌋

Rationale for the existence of [constr_5059]: This ensures that all MetaDataItems will be properly aligned without any padding between individual MetaDataItems.

[TPS_ECUC_06086] Relevance of the order of MetaDataItems of an MetaDataType ⌈The order of MetaDataItems of an MetaDataType defines the order and position of the meta data items in the meta data array of the respective Pdu.⌋

3.4 COM-Stack configuration

To cope with the complexity of the COM-Stack configuration, reoccurring patterns have been applied which will be described in this section. Only the patterns, together with some examples, are shown. To get detailed specification of the configuration for each individual module please refer to the actual BSW SWS documents of these modules.

3.4.1 Handle IDs

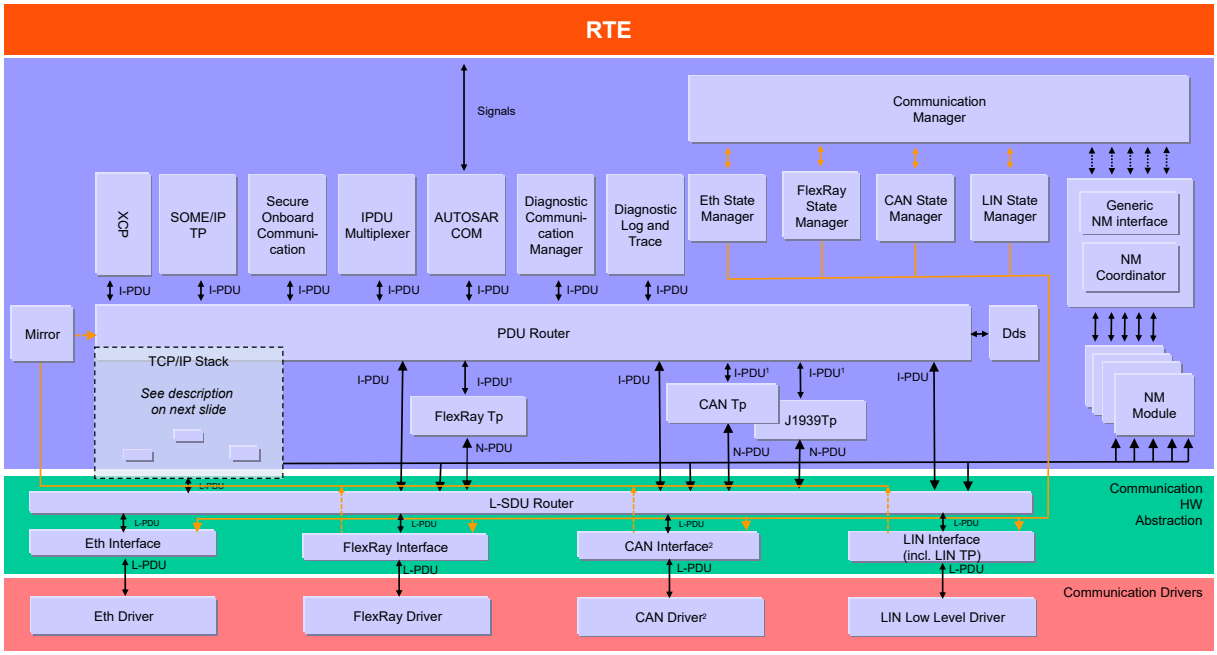

Figure 3.13: Interfaces in the COM-Stack [12] (page id: 3hd8w)

In figure 3.13 a detailed view of the COM-Stack modules and their interaction is shown. There are several kinds of interactions between adjacent[3] modules.

3.4.1.1 Handle ID concept

The API definitions in the COM-Stack utilize two concepts to achieve the interaction between adjacent modules:

- Pointers to Pdu data buffer (the Pdu data buffer contains the actual communicated information, depending on the actual layer the interaction happens)
- Handle IDs to identify to what Pdu the pointer is referring to.

A typical API call is for instance:

```
PduR_ComTransmit(PduIdType ComTxPduId, PduInfoType *PduInfoPtr)
```

Which BSW Module is actually providing the value of the Handle ID is specified in the `ECU Configuration Parameter Definition` of the corresponding BSW Module (see section 3.4.1.2 for details on the specification).

The choice of the value for a Handle ID is open to the implementation of the providing module. There might be different strategies to optimize the Handle ID values and therefore the internal structures of the implementation may have an influence on the choice of the values.

Also the Handle IDs can be chosen freely per module, so a Pdu might be sent from Com to the PduR with the `ID=5` and then the PduR transmits it further to the CanIf with `ID=19`. In the configuration information of the PduR it has to be possible to conclude that if a Pdu arrives from Com with `ID=5` it has to be forwarded to the CanIf with `ID=19`.

It has to be guaranteed that each Pdu does have a unique handle ID within the scope of the corresponding API. For example: The PduR gets transmission requests from both, the Com and the Dcm modules. But there are also two distinct APIs defined for those requests:

- `PduR_ComTransmit(...)`
- `PduR_DcmTransmit(...)`

Therefore the PduR can distinguish two Pdus, even when they have the same handle ID but are requested via different APIs.

Another use-case in the COM-Stack only provides one API for all the callers: the interface layer (CanIf, FrIf, LinIf).

- `CanIf_Transmit(...)`

[3]Modules are called adjacent if they share an interface, so PduR and Com are adjacent, while PduR and Can driver are not.

Here it has to be guaranteed that each transmit request for a distinct Pdu does have a unique handle ID.

The actual values of the handle IDs can only be assigned properly when the configuration of one module is completed, since only then the internal data structures can be defined.

In the next sections the patterns used to define and utilize Handle IDs are described.

3.4.1.2 Definition of Handle IDs

Handle IDs are defined by the module providing the API and used by the module calling the API. Handle IDs that are used in callback functions (e.g. Tx Confirmation functions or Trigger Transmit functions) shall be defined by the upper layer module. In the upper layer module the same HandleId shall be used for the Tx Confirmation and for the Trigger Transmit callback functions. I.e. the module that receives a transmission request can call the Tx confirmation callback with a different Handle Id than the transmission request Handle Id. This is a difference to previous releases of AUTOSAR where the Tx confirmation was called with the same Handle Id.

The `ECU Configuration Values` (which holds the actual values of configuration parameters) is structured according to the individual BSW Module instances. Therefore the `ECU Configuration Parameter Definition` is also structured in this way.

In figure 3.14 an exemplary definition of a partial Can Interface transmit configuration is shown.

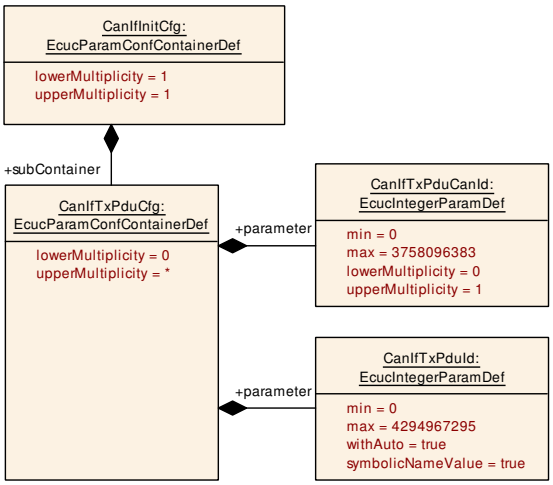

Figure 3.14: Example of Can Interface Tx configuration

The configuration of the module `CanIf` may contain several `CanIfTxPduConfig` objects.

Each `CanIfTxPduConfig` object contains information on one Pdu which is coming from an upper layer (e.g. PduR or Nm) and is going to some Can driver. In this example the `CanIfCanTxPduCanId` and `CanIfCanTxPduDlc` are specified for each

to be transmitted Pdu. There is a similar structure needed for the receive use-case as well.

Additionally the parameter `CanIfCanTxPduId` is specified. This integer parameter will later hold the actual value for the handle ID. So the handle ID value is stored inside the structure of the defining module.

Since the handle ID `CanIfCanTxPduId` is part of the container `CanIfTxPduConfig` the semantics of the symbolic names can be applied.

The described example only applies for the communication between CanIf and Upper Layer modules. CanDrv does not support the handle ID concept and indicates TxConfirmation using the PduId passed during Can_Write().

[TPS_ECUC_02106] Handle Id which needs to be shared between several modules ⌈ If a configuration parameter holds a handle Id which needs to be shared between several modules it shall have the `symbolicNameValue` = true set.⌋

Thus it is required that all handle Id values are accessible via a symbolic name reference (see section 3.4.1.4).

3.4.1.3 Agreement on Handle IDs

During the configuration of a module, information for each `Pdu` flowing through this module is created (see again figure 3.14: `CanIfTxPduConfig`) which hold module-specific configuration information. Now each of these "local" `Pdu` configurations needs to be related to a "global" `Pdu` element (see section 3.3.7) representing information flowing through the COM-Stack. This is done by introducing a `EcucReferenceDef` from the "local" `Pdu` to the "global" `Pdu`.

In figure 3.15 this relationship is shown for the `PduRDestPdu` and the `CanIfTxPduConfig`.

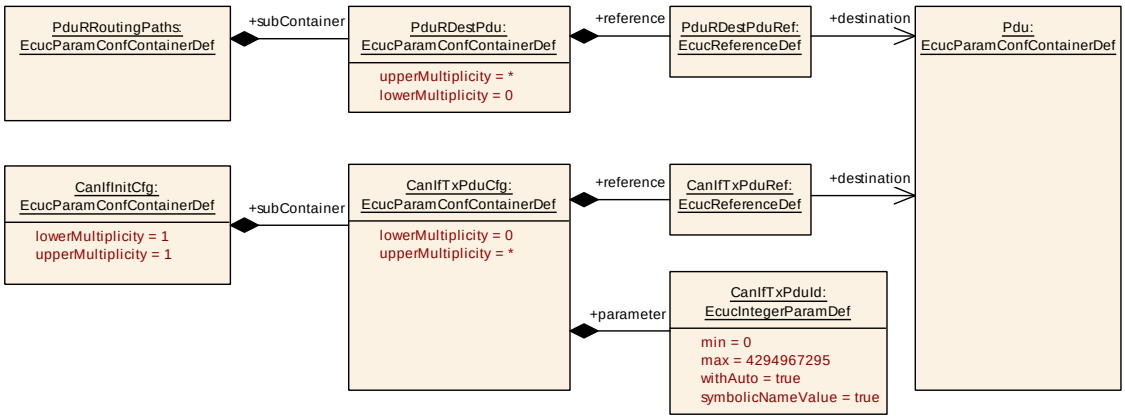

Figure 3.15: Transmission from PduR to CanIf

There are two reasons why the "global" `Pdu` has been introduced and why all "local" Pdus have to point to the "global" `Pdu` only.

- When doing the configuration of module PduR only the "global" Pdu needs to be present, there is no need for the "local" Pdu in the CanIf to be present yet.
- The References are stored in the "local" Pdu structure, so changes applied do only influence the structure of the changed module.

Taking the structure shown in figure 3.15 it is now possible to generate both modules.

The CanIf (automatic) configuration editor collects all "local" CanIfTxPduConfigs and generates/stores the values for their handle ID in CanIfCanTxPduId. If the CanIf needs to know where the Pdu transmit request is coming from it can follow the PduIdRef to the "global" Pdu and then "query" all references pointing to that Pdu. By following those references in reversed direction the transmitting module can be found.

The PduR generator has to know which handle ID to use for each Pdu that has to be sent to the CanIf. To get the actual handle ID value the mechanism is the same in the CanIf use-case: follow the "global" Pdu reference and "query" the modules pointing to that "global" Pdu. Then find the module(s) type this Pdu is going to be transmitted to. In case of a multicast there might be several modules to send the same Pdu to.

With this approach a high degree of decoupling has been achieved between the configuration information of the involved modules. Even when modules are adjacent and need to share information like handle ID, the references between the modules are always indirect using the "global" Pdu elements.

3.4.1.4 Handle IDs with symbolic names

The usage of handle Ids together with symbolic names is targeting several use-cases for the methodology of configuring adjacent modules. For the definition of possible configuration approaches please refer to section B.1.1.

For the discussion of the Handle Id use-cases two basic approaches can be distinguished when dividing the methodology into the steps configuration editing and module generation:

- Handle Ids assigned by the configuration editor
- Handle Ids assigned by the module generator

It is assumed that the configuration and generation of the whole stack is done using different tools (possibly from different vendors) which might implement one of the two approaches mentioned above.

In order to support the definition whether a parameter value shall be provided by the user or whether it will be calculated by the editor / generator tooling the attribute withAuto has been introduced to the EcucParameterDef (see section 2.3.5).

In requirement [TPS_ECUC_02106] it is required that all handle Ids are represented as symbolicNameValue = true configuration parameters thus decoupling the value from its usage.

In requirement [TPS_ECUC_02107] it is required that the assigned values are stored in the XML (latest after module generation) so the assigned values are documented. In case the assignment of values has to be performed at a later point in time again (with updated input information) the non affected values can be preserved. It is also needed to support debugging.

In requirement [TPS_ECUC_02108] it is required that the handle Id values are always generated into the module's header file. With this approach it is possible to freely choose the configuration approach of the adjacent modules.

This approach has significant effect on the methodology due to the circular dependencies between the adjacent modules(Com sends to the PduR using PduR handle Ids, PduR indicates to Com using Com handle Ids). Therefore the configuration of all adjacent modules has to be re-visited in case some handle Id changes happen. This contributes to the approach that FIRST the *configuration* of the stack is performed and SECOND the *generation* is triggered.

An example of this approach is provided below: By adding the attribute `symbolicNameValue` = true to the parameter holding the handle ID (in figure 3.15 this is the parameter `CanIfTxPduId`) the code generator doing the `CanIf` will generate a `#define` in the `CanIf_cfg.h` file.

According to [TPS_ECUC_02108] the name of the symbol is composed of the module abbreviation <MA> of the declaring BSW Module followed by the literal "Conf_" followed by the shortName of the EcucParamConfContainerDef of the declaring module followed by the shortName of the EcucContainerValue container which holds the symbolicNameValue configuration parameter value. The value is the actual number assigned to that handle ID.

For example in `CanIf_cfg.h`:

```
#define CanIfConf_CanIfTxPduCfg_Pdu_2345634_985 17
```

The benefit is that the generator of the `PduR` does not need to wait for the `CanIf` to be configured completely and handle IDs are generated. If the `CanIf` publishes the symbolic names for the handle IDs, the `PduR` can expect those symbolic names and generate the `PduR` code using those symbolic names.

For example in `PduR.c`:

```
CanIf_Transmit( CanIfConf_CanIfTxPduCfg_Pdu_2345634_985, PduPtr
)
```

Therefore the `PduR` can be generated as soon as its own configuration is finished and there is no need to wait for the `CanIf` to be finished completely. However, at least the "local" `Pdu` in the `CanIf` has to be already created to allow this, because the name of the symbol has to be fetched from this configuration.

Of course the `PduR` can only be compiled after the `CanIf` has been generated as well, but with the utilization of the symbolic names together with handle IDs an even higher degree of decoupling in the configuration process is achieved.

3.4.2 Configuration examples for the Pdu Router

In this section several use-cases of the PduR are described from the configuration point of view. The focus is on the interaction of the PduR configuration with the configuration of the other COM-Stack modules. Therefore only some configuration parameters are actually shown in these examples.

3.4.2.1 Tx from Com to CanIf

In the example in figure 3.16 a Pdu is sent from the Com module – via the Pdu Router – to the Can Interface. Since this one Pdu is handed over through these layers there is only need for one global Pdu object `System_Pdu`.

The Com module's configuration points to the `System_Pdu` to indicate which Pdu shall be sent. The actual Handle Id which has to be used in the API call will however be defined by the PduR in the parameter `PduRSrcPdu::HandleId`. In this example the Com module has to use the Handle Id `23` to transmit this Pdu to the PduR.

Then, since the CanIf is pointing to the same `System_Pdu` the PduR can be configured to send this Pdu to the CanIf. The Handle Id is defined in the CanIf configuration in the parameter value of `CanIfCanTxPduId`.

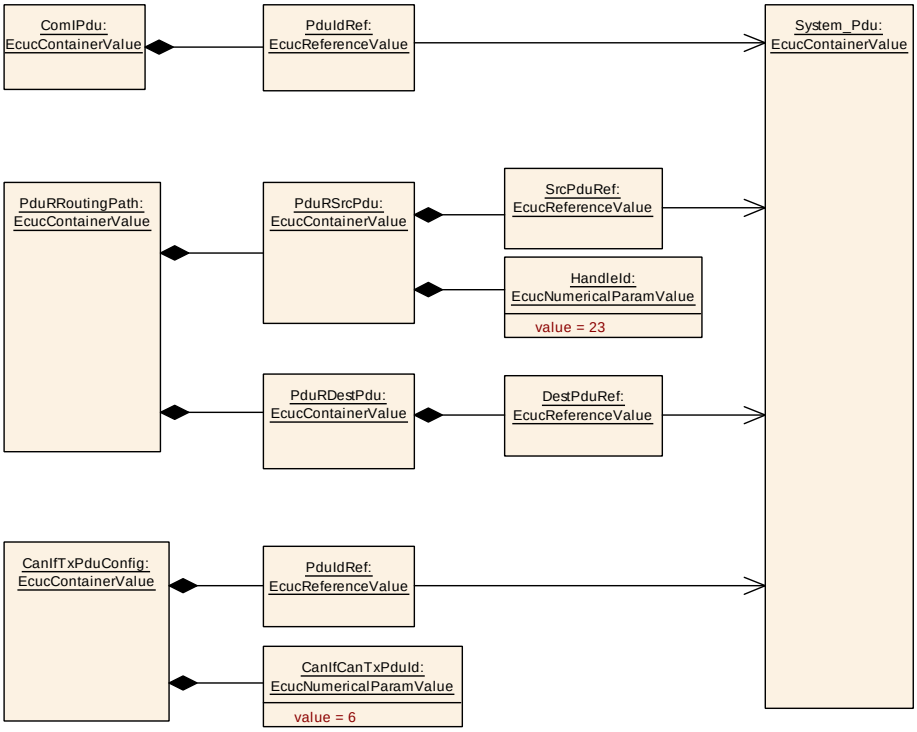

Figure 3.16: Tx from Com to CanIf example

3.4.2.2 Rx from CanIf to Com

In the example in figure 3.17 the reception use-case from the CanIf to the Com module is configured. Here the Handle Ids are defined in the PduR and the Com module's configuration.

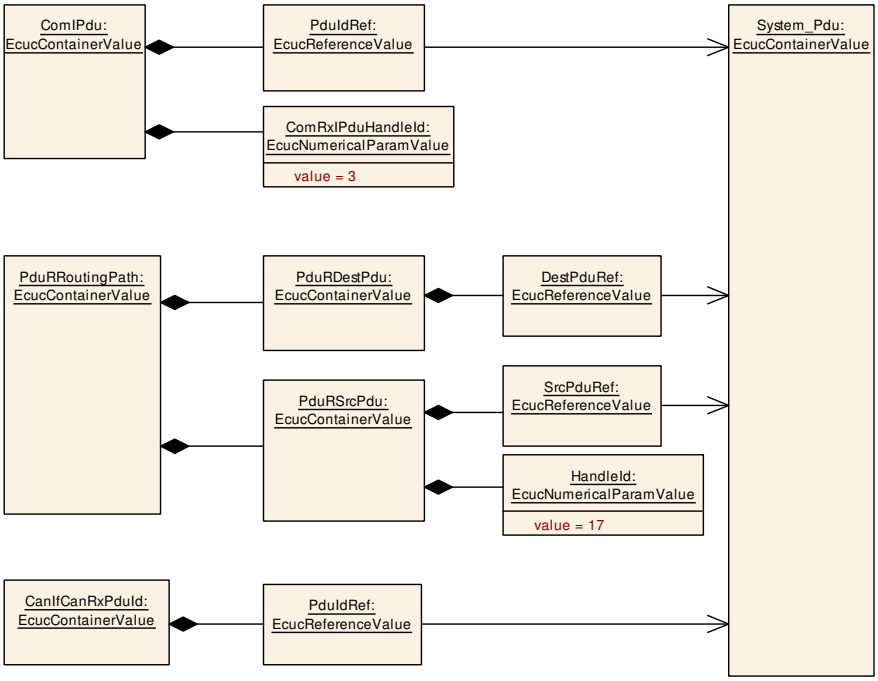

Figure 3.17: Rx from CanIf to Com example

3.4.2.3 Gateway from CanIf to FrIf

In the example in figure 3.18 the gateway use-case is shown. Since there are two Pdus involved there are two `System_Pdu` objects defined: one which is representing the Can Pdu and one which represents the Fr Pdu. Via the references to these two `System_Pdu` objects the gateway is configured.

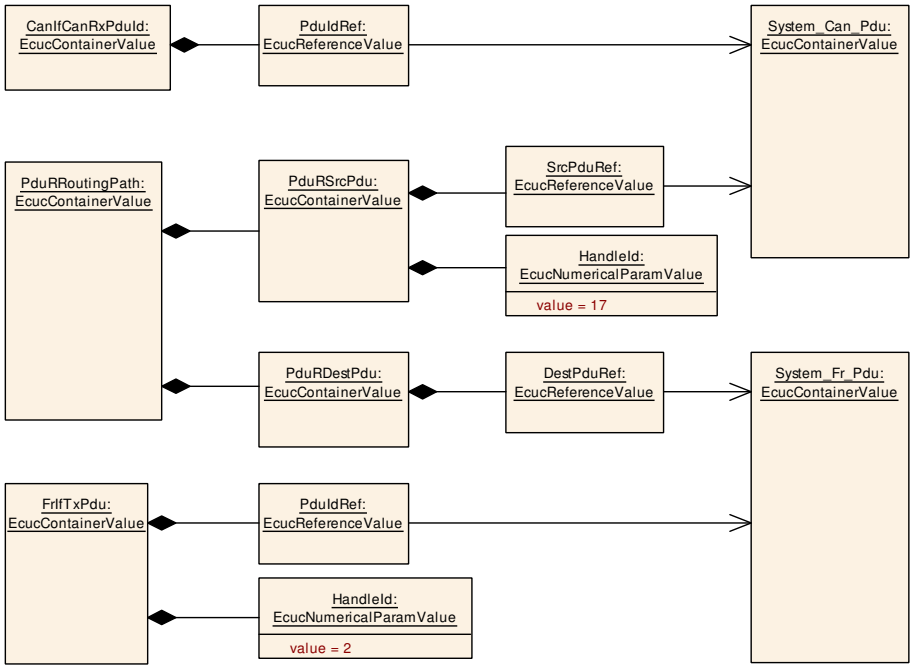

Figure 3.18: Gateway from CanIf to FrIf example

3.4.3 Communication Channel IDs

For the configuration of the control path modules (e.g. Communication manager, state managers, network managers) the respective channels are identified using a unique *Communication Channel ID* approach. This is different than the configuration of the *Pdu Handle ID*s of the COM-Stack (see section 3.4.1) where individual *Pdu Handle ID*s are configured per module.

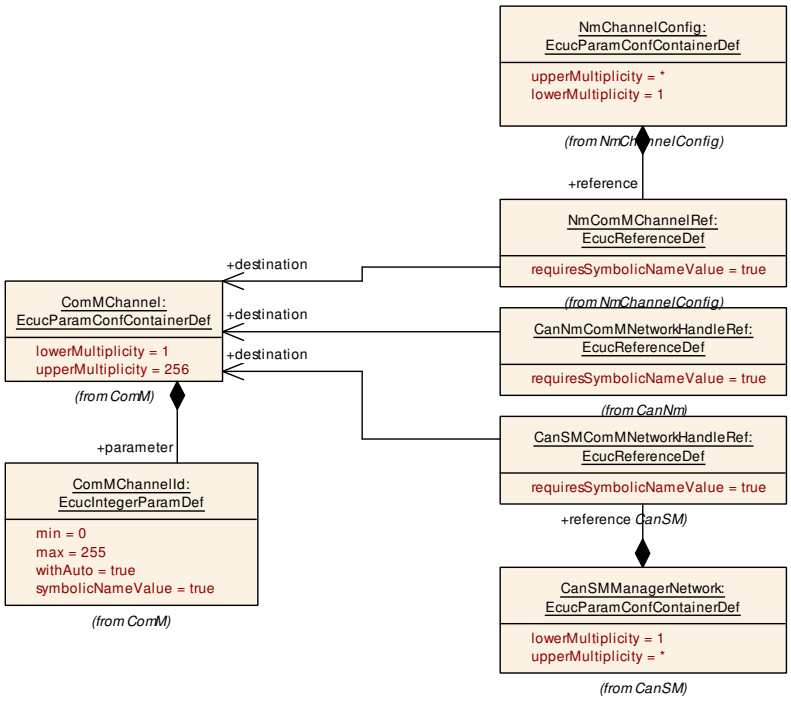

Figure 3.19: Example of Channel interaction between ComM, Can and Nm

In figure 3.19 the `ComMChannel` defines a global communication channel and provides the Communication Channel ID of this channel in the parameter value `ComMChannelId`. Other modules using communication channels (e.g. Nm, CanSM, CanNm, ...) refer to the `ComMChannel` and can utilize the Communication Channel ID in two ways:

- the module does not store the value of the Communication Channel ID itself but always relies on the value provided by the ComM module (like shown for CanNm).
- the module replicates the value of the Communication Channel ID and requires that the replicated id value is equal to the one provided by ComM module (like shown for Nm and CanSM).

Both approaches are currently used in the COM-Stack configuration.

3.4.4 Copy Configuration for PDUs

As described in Chapter 3.3.7 the "virtual" module `EcuC` in the ECU Configuration collects configuration per Pdu. Those Pdus do not belong to an individual module, but can be referenced by any module to signal their usage. With this approach, a Pdu flowing through the COM-Stack can be modeled. A lower layer module could reference a Pdu as receiving upper layer Pdu. The next upper layer module may reference the same Pdu as receiving lower layer Pdu. This upper layer module may have a Pdu forwarding configured, where the lower layer Pdu is linked to the upper layer Pdu of the next upper layer (and so on...). Thus, the Pdu flow from the lower layer module across all intermediate layer modules to the destination module, and vice versa, can

be configured. For some use cases, it is necessary that the configuration parameters of Pdus, which forms a Pdu flow, have the same value configured.

The configuration parameter KeepLocalPduBuffer indicates that a module, which processes the Pdu and produces specific data (e.g. protocol header) for this Pdu, which is locally kept by the module, should keep the produced data until it is indicated to release the allocated local buffer (TxConfirmation / ReleaseRxBuffer). This could be used to perform hardware supported data transfer a.k.a. DMA-copy operation, where the destination module could trigger an asynchronous data transfer and indicate the release of the local buffer as soon as the transfer is finalized. See [16, BSW General] Chapter 7.1.21 *"Asynchronous copy approach (DRAFT)"* for more details.

[constr_3793] Usage of KeepLocalPduBuffer

Status: DRAFT
Imposition time: CP: IT_EcucVal

⌈All Pdus that belong to the same Pdu flow shall have KeepLocalPduBuffer either set to TRUE or set to FALSE.⌋

The configuration parameter PduBufferAlignment indicates that the alignment of produced data stored in a local buffer respect the configured buffer alignment required by the underlying hardware platform.

[constr_3794] Usage of PduBufferAlignment

Status: DRAFT
Imposition time: CP: IT_EcucVal

⌈All Pdus that belong to the same Pdu flow shall have PduBufferAlignment set to the same value.⌋

This hardware supported data transfer only makes sense if the number of bytes to copy surmounts a certain level, which is strongly dependent on the microcontroller design, and therefore configurable (per core) as EcucCoreDefinition.EcucDmaThreshold.

3.5 CDD module

The CDD module describes the minimal requirements that are necessary for the configuration of a Complex Driver with respect to the surrounding standardized BSW modules.

[TPS_ECUC_06031] Interaction of Complex Driver with standardized AUTOSAR BSW modules ⌈If a Complex Driver wants to interact with a surrounding standardized BSW module it has to define a Vendor Specific Module Definition from the Standardized CDD Module Definition. The rules that shall be followed when generating the Vendor Specific Module Definition are described by [TPS_ECUC_06038].⌋

As defined in [TPS_ECUC_06001] the shortName of a VSMD module shall be the same as the shortName of the StMD. According to this requirement the shortName of the module definition of a Complex Driver is always "Cdd".

[TPS_ECUC_06036] Distinction of module definitions of Complex Drivers ⌈To distinguish module definitions of Complex Drivers from each other the package structure shall be used.⌋

[TPS_ECUC_06037] apiServicePrefix attribute for Complex Driver module and Xfrm module ⌈The module abbreviation of a Complex Driver and Xfrm Module shall be equal to its apiServicePrefix attribute.⌋

[constr_3023] Usage of apiServicePrefix

Imposition time: CP: IT_EcucVal

⌈The attribute apiServicePrefix is mandatory for VSMDs derived from the CDD and Xfrm StMD. The attribute shall not be provided for VSMDs derived from any other StMDs.⌋

Consider a Complex Driver named "MyCdd". The VSMD of this Complex Driver has to be derived from the CDD StMD. The shortName of the module definition of this Complex Driver has to be equal to "Cdd". The apiServicePrefix attribute is mandatory for the VSMD of this Complex Driver and has to be equal to "MyCdd".

Note that the configuration parameters for the VSMD of CDD do not specify any configuration class. It is up to the implementor of the specific CDD to define the configuration class for all configuration parameters - standardized and vendor specific ones (see [TPS_ECUC_02139]).

[ECUC_Cdd_00016] Definition of EcucModuleDef Cdd ⌈

Module Name	Cdd
Description	The CDD module describes the minimal requirements that are necessary for the configuration of a CDD with respect to the surrounding standardized BSW modules.
Post-Build Selectable Support	true
Post-Build Loadable Support	true

Included Containers		
Container Name	**Multiplicity**	**Description**
CddComStackContribution	0..1	Contribution of COM Stack modules.
CddEcucPartitionInteraction	0..1	This optional container holds the partition interaction configuration.
CddGeneral	0..1	Contains the general configuration parameters of the module.
CddGlobalTimeContribution	0..1	Contribution of Global Time modules.

⌋

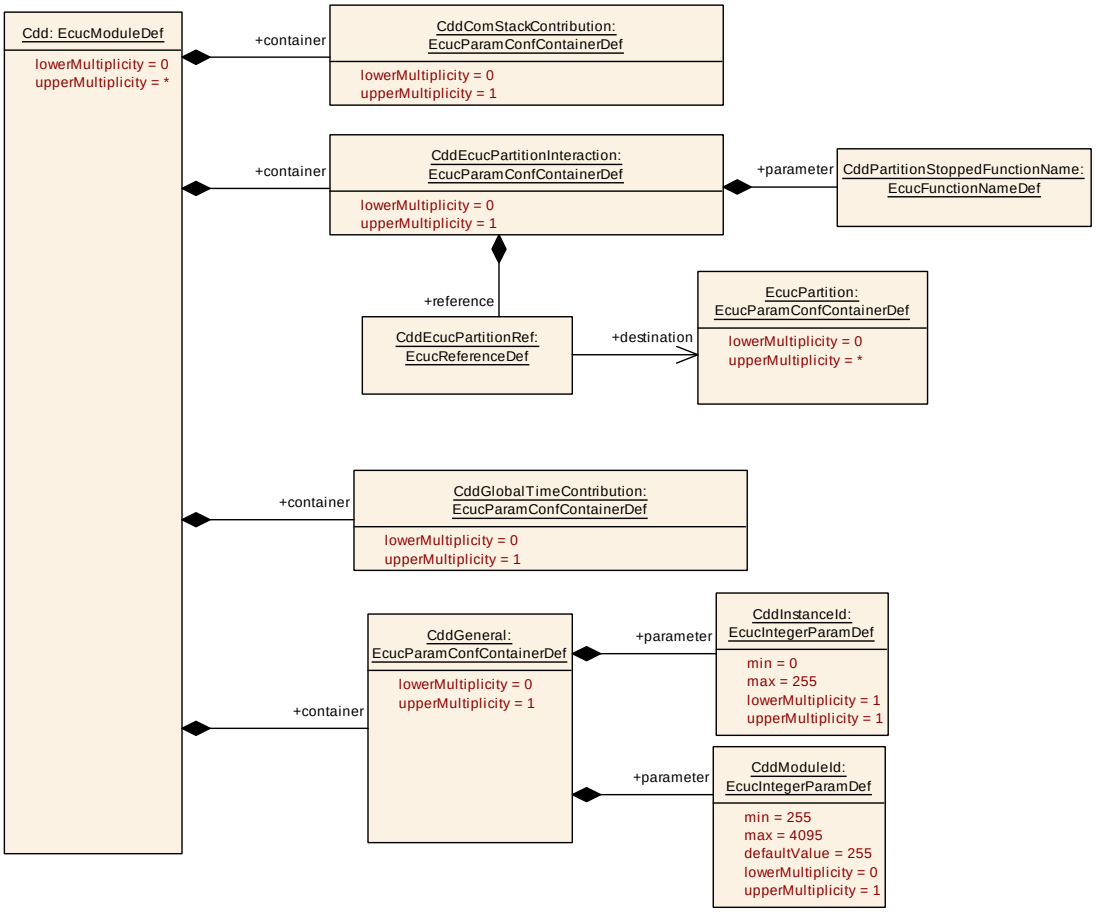

Figure 3.20: Cdd Module

[ECUC_Cdd_00083] Definition of EcucParamConfContainerDef CddGeneral ⌈

Container Name	CddGeneral
Parent Container	Cdd
Upstream mapping defined	–
Description	Contains the general configuration parameters of the module.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
CddInstanceId	1	[ECUC_Cdd_00084]
CddModuleId	0..1	[ECUC_Cdd_00085]

No Included Containers

⌋

[ECUC_Cdd_00084] Definition of EcucIntegerParamDef CddInstanceId ⌈

Parameter Name	CddInstanceId	
Parent Container	CddGeneral	
Upstream mapping defined	–	
Description	Specifies the InstanceId of this module instance. If only one instance is present it shall have the Id 0.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 255	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_Cdd_00085] Definition of EcucIntegerParamDef CddModuleId ⌈

Parameter Name	CddModuleId	
Parent Container	CddGeneral	
Upstream mapping defined	–	
Description	Specifies the ModuleId of this module.	
Multiplicity	0..1	
Type	EcucIntegerParamDef	
Range	255 .. 4095	
Default value	255	
Pre-Compile only	true	

⌋

[constr_5108] `CddModuleId` range restriction

Imposition time: CP: `IT_EcucVal`

⌈The range of `CddModuleId` is restricted to the value 255 and to the range of values 2048..4095.⌋

[ECUC_Cdd_00038] Definition of EcucParamConfContainerDef CddEcucPartitionInteraction ⌈

Container Name	CddEcucPartitionInteraction
Parent Container	Cdd
Upstream mapping defined	–
Description	This optional container holds the partition interaction configuration.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
CddPartitionStoppedFunctionName	1	[ECUC_Cdd_00040]
CddEcucPartitionRef	1	[ECUC_Cdd_00039]

No Included Containers

⌋

[ECUC_Cdd_00040] Definition of EcucFunctionNameDef CddPartitionStoppedFunctionName ⌈

Parameter Name	CddPartitionStoppedFunctionName
Parent Container	CddEcucPartitionInteraction
Upstream mapping defined	–
Description	Function name to be called when the partition which is triggering the Complex Driver is stopped.
Multiplicity	1
Type	EcucFunctionNameDef
Default value	–
Regular Expression	–
Pre-Compile only	true

⌋

[ECUC_Cdd_00039] Definition of EcucReferenceDef CddEcucPartitionRef ⌈

Parameter Name	CddEcucPartitionRef
Parent Container	CddEcucPartitionInteraction
Upstream mapping defined	–
Description	Reference to the "EcucPartition" which executes the software which triggers the CDD.
Multiplicity	1
Type	Reference to EcucPartition
Pre-Compile only	true

⌋

[ECUC_Cdd_00017] Definition of EcucParamConfContainerDef CddComStackContribution ⌈

Container Name	CddComStackContribution
Parent Container	Cdd
Upstream mapping defined	–
Description	Contribution of COM Stack modules.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

No Included Parameters

Included Containers		
Container Name	**Multiplicity**	**Description**
CddComIfUpperLayerContribution	0..1	Parameters that are necessary for the configuration of a Complex Driver that serves as the UpperLayer of the Com Interface module.

▽

△

Included Containers		
Container Name	**Multiplicity**	**Description**
CddComMLowerLayerContribution	0..1	Parameters that are necessary for the configuration of a Complex Driver that serves as the LowerLayer of the Communication Manager module.
CddGenericNmLowerLayer Contribution	0..1	Parameters that are necessary for the configuration of a Complex Driver that serves as the LowerLayer of the Generic NM module.
CddJ1939RmContribution	0..1	Contribution of J1939Rm module
CddLSduRLowerLayerContribution	0..1	Parameters that are necessary for the configuration of a Complex Driver that serves as the LowerLayer of the L-Sdu Router module. **Tags:** atp.Status=draft
CddLSduRUpperLayerContribution	0..1	Parameters that are necessary for the configuration of a Complex Driver that serves as the UpperLayer of the L-Sdu Router module. **Tags:** atp.Status=draft
CddPduRLowerLayerContribution	0..1	Parameters that are necessary for the configuration of a Complex Driver that serves as the LowerLayer of the Pdu Router module.
CddPduRUpperLayerContribution	0..1	Parameters that are necessary for the configuration of a Complex Driver that serves as the UpperLayer of the Pdu Router module.
CddSoAdUpperLayerContribution	0..1	Parameters that are necessary for the configuration of a Complex Driver that serves as the UpperLayer of the SoAd module.

⌋

The following sections describe particular COM stack modules and the interaction with Complex Drivers.

3.5.1 Pdu Router

In the AUTOSAR COM Stack upper and lower layer Complex Drivers are allowed to access the Pdu Router. In both cases the Pdus that are exchanged between the CDD and the Pdu Router shall be configured. The contribution of the Complex Driver implies a reference to the global Pdu and the definition of a HandleId. Figure 3.21 shows an example of a Complex Driver between the CanIf and the PduR and one Complex Driver above the PduR.

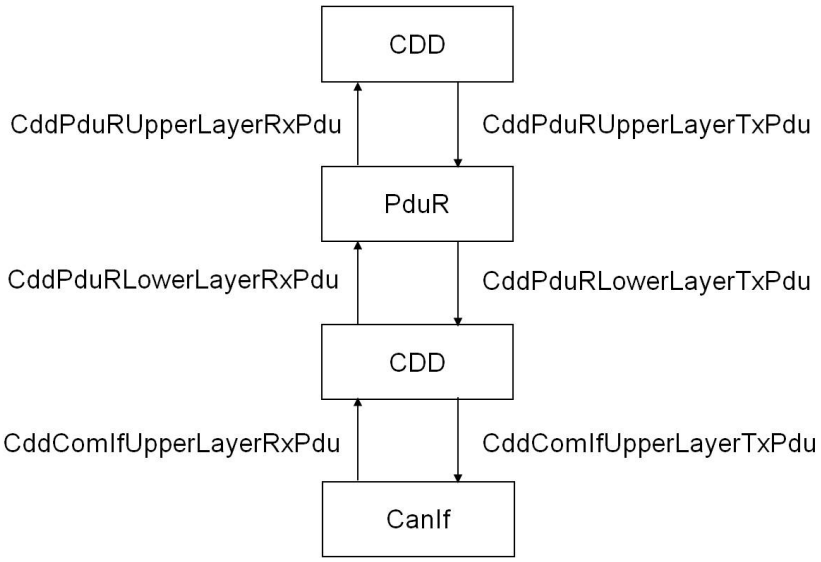

Figure 3.21: CDD Example

Figure 3.22 shows the CDD contribution in the configuration model.

Note that the optional presence of the *TxPdu* and *RxPdu* does not influence the existence of the respective APIs in the *Cdd*.

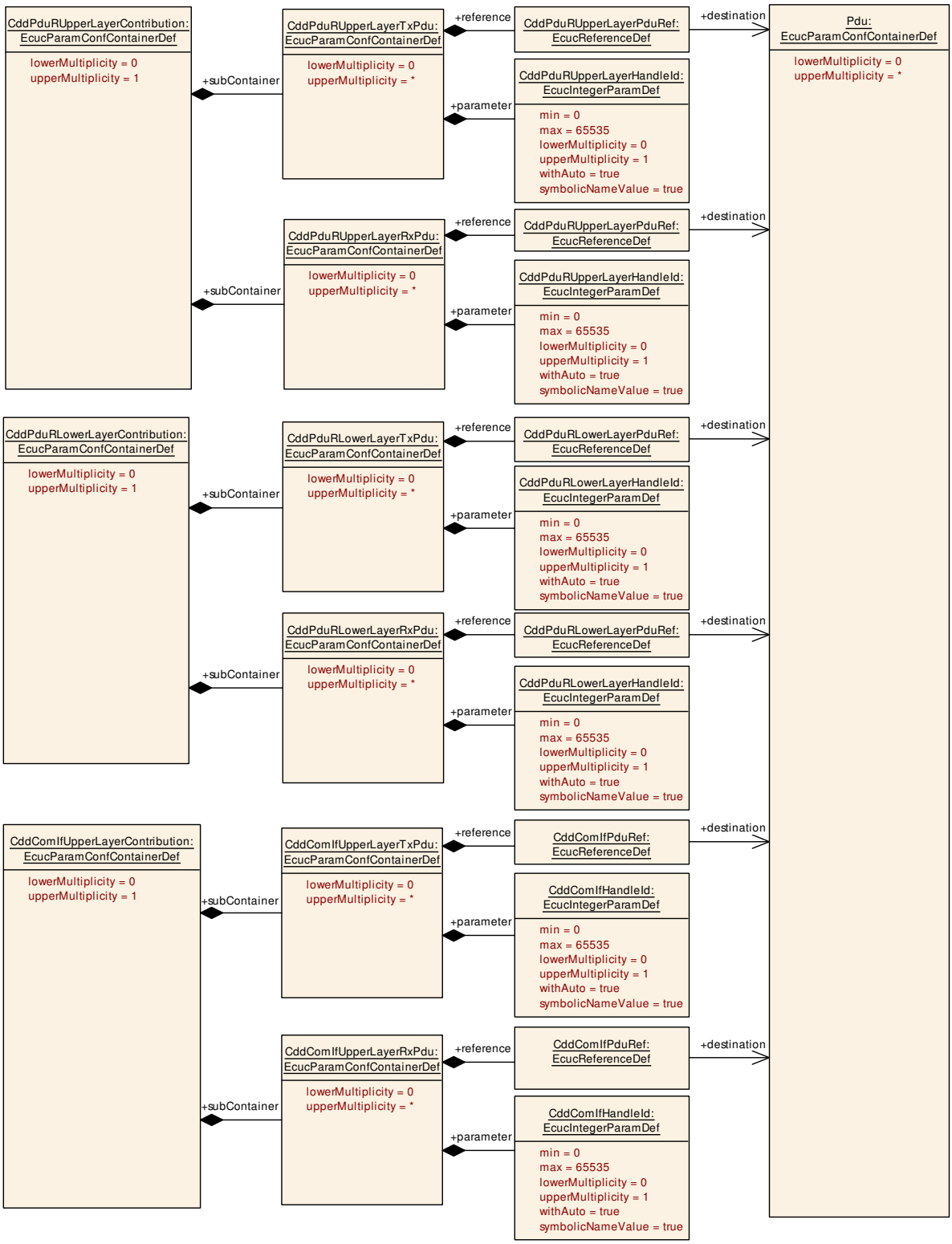

Figure 3.22: PduR and Com Interface contribution

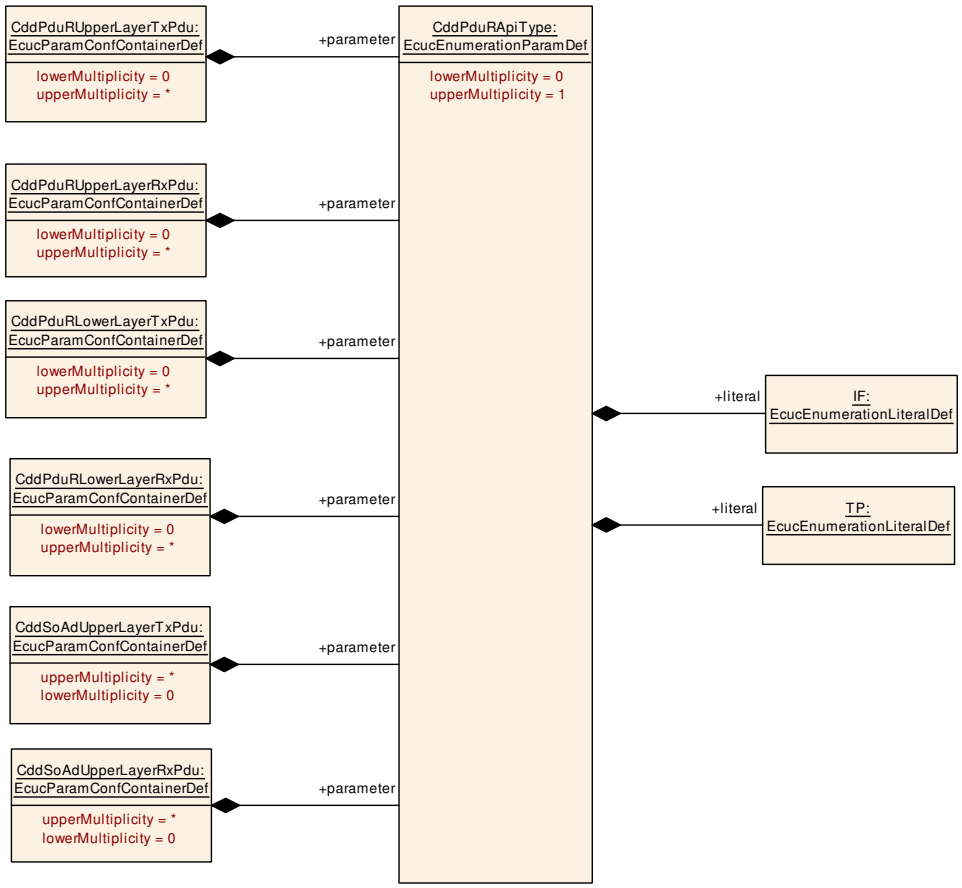

Figure 3.23: Configuration of the CDD interface API type

[ECUC_Cdd_00026] Definition of EcucParamConfContainerDef CddPduRUpperLayerContribution ⌈

Container Name	CddPduRUpperLayerContribution
Parent Container	CddComStackContribution
Upstream mapping defined	–
Description	Parameters that are necessary for the configuration of a Complex Driver that serves as the UpperLayer of the Pdu Router module.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

No Included Parameters

Included Containers		
Container Name	**Multiplicity**	**Description**
CddPduRUpperLayerRxPdu	0..*	This container specifies Rx PDUs that are exchanged between the CDD and the standardized BSW module.
CddPduRUpperLayerTxPdu	0..*	This container specifies Tx PDUs that are exchanged between the CDD and the standardized BSW module.

⌋

[ECUC_Cdd_00022] Definition of EcucParamConfContainerDef CddPduRLowerLayerContribution ⌈

Container Name	CddPduRLowerLayerContribution
Parent Container	CddComStackContribution
Upstream mapping defined	–
Description	Parameters that are necessary for the configuration of a Complex Driver that serves as the LowerLayer of the Pdu Router module.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

No Included Parameters

Included Containers		
Container Name	**Multiplicity**	**Description**
CddPduRLowerLayerRxPdu	0..*	This container specifies Rx PDUs that are exchanged between the CDD and the standardized BSW module.
CddPduRLowerLayerTxPdu	0..*	This container specifies Tx PDUs that are exchanged between the CDD and the standardized BSW module.

⌋

[ECUC_Cdd_00027] Definition of EcucParamConfContainerDef CddPduRUpperLayerTxPdu ⌈

Container Name	CddPduRUpperLayerTxPdu
Parent Container	CddPduRUpperLayerContribution
Upstream mapping defined	–
Description	This container specifies Tx PDUs that are exchanged between the CDD and the standardized BSW module.
Multiplicity	0..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
CddPduRApiType	0..1	[ECUC_Cdd_00052]
CddPduRUpperLayerHandleId	0..1	[ECUC_Cdd_00029]
CddPduRUpperLayerPduRef	1	[ECUC_Cdd_00028]

No Included Containers

⌋

For parameter table [ECUC_Cdd_00052] CddPduRApiType, see definition below container CddPduRLowerLayerRxPdu.

[ECUC_Cdd_00029] Definition of EcucIntegerParamDef CddPduRUpperLayer HandleId ⌈

Parameter Name	CddPduRUpperLayerHandleId
Parent Container	CddPduRUpperLayerTxPdu
Upstream mapping defined	–
Description	ECU wide unique, symbolic handle for the Pdu.
Multiplicity	0..1
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)
Range	0 .. 65535
Default value	–
With Auto	true
Pre-Compile only	true

⌋

[ECUC_Cdd_00028] Definition of EcucReferenceDef CddPduRUpperLayerPdu Ref ⌈

Parameter Name	CddPduRUpperLayerPduRef
Parent Container	CddPduRUpperLayerTxPdu
Upstream mapping defined	–
Description	Reference to the "global" Pdu structure to allow harmonization of handle IDs in the COM-Stack.
Multiplicity	1
Type	Reference to Pdu
Pre-Compile only	true

⌋

[ECUC_Cdd_00043] Definition of EcucParamConfContainerDef CddPduRUpper LayerRxPdu ⌈

Container Name	CddPduRUpperLayerRxPdu
Parent Container	CddPduRUpperLayerContribution
Upstream mapping defined	–
Description	This container specifies Rx PDUs that are exchanged between the CDD and the standardized BSW module.
Multiplicity	0..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
CddPduRApiType	0..1	[ECUC_Cdd_00052]
CddPduRUpperLayerHandleId	0..1	[ECUC_Cdd_00045]
CddPduRUpperLayerPduRef	1	[ECUC_Cdd_00044]

No Included Containers

⌋

For parameter table [ECUC_Cdd_00052] CddPduRApiType, see definition below container CddPduRLowerLayerRxPdu.

[ECUC_Cdd_00045] Definition of EcucIntegerParamDef CddPduRUpperLayerHandleId ⌈

Parameter Name	CddPduRUpperLayerHandleId
Parent Container	CddPduRUpperLayerRxPdu
Upstream mapping defined	–
Description	ECU wide unique, symbolic handle for the Pdu.
Multiplicity	0..1
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)
Range	0 .. 65535
Default value	–
With Auto	true
Pre-Compile only	true

⌋

[ECUC_Cdd_00044] Definition of EcucReferenceDef CddPduRUpperLayerPduRef ⌈

Parameter Name	CddPduRUpperLayerPduRef
Parent Container	CddPduRUpperLayerRxPdu
Upstream mapping defined	–
Description	Reference to the "global" Pdu structure to allow harmonization of handle IDs in the COM-Stack.
Multiplicity	1
Type	Reference to Pdu
Pre-Compile only	true

⌋

[ECUC_Cdd_00023] Definition of EcucParamConfContainerDef CddPduRLowerLayerTxPdu ⌈

Container Name	CddPduRLowerLayerTxPdu
Parent Container	CddPduRLowerLayerContribution
Upstream mapping defined	–
Description	This container specifies Tx PDUs that are exchanged between the CDD and the standardized BSW module.
Multiplicity	0..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
CddPduRApiType	0..1	[ECUC_Cdd_00052]
CddPduRLowerLayerHandleId	0..1	[ECUC_Cdd_00025]
CddPduRLowerLayerPduRef	1	[ECUC_Cdd_00024]

No Included Containers

⌋

For parameter table [ECUC_Cdd_00052] CddPduRApiType, see definition below container CddPduRLowerLayerRxPdu.

[ECUC_Cdd_00025] Definition of EcucIntegerParamDef CddPduRLowerLayerHandleId ⌈

Parameter Name	CddPduRLowerLayerHandleId	
Parent Container	CddPduRLowerLayerTxPdu	
Upstream mapping defined	–	
Description	ECU wide unique, symbolic handle for the Pdu.	
Multiplicity	0..1	
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)	
Range	0 .. 65535	
Default value	–	
With Auto	true	
Pre-Compile only	true	

⌋

[ECUC_Cdd_00024] Definition of EcucReferenceDef CddPduRLowerLayerPduRef ⌈

Parameter Name	CddPduRLowerLayerPduRef
Parent Container	CddPduRLowerLayerTxPdu
Upstream mapping defined	–
Description	Reference to the "global" Pdu structure to allow harmonization of handle IDs in the COM-Stack.
Multiplicity	1
Type	Reference to Pdu
Pre-Compile only	true

⌋

[ECUC_Cdd_00046] Definition of EcucParamConfContainerDef CddPduRLowerLayerRxPdu ⌈

Container Name	CddPduRLowerLayerRxPdu
Parent Container	CddPduRLowerLayerContribution
Upstream mapping defined	–
Description	This container specifies Rx PDUs that are exchanged between the CDD and the standardized BSW module.
Multiplicity	0..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
CddPduRApiType	0..1	[ECUC_Cdd_00052]
CddPduRLowerLayerHandleId	0..1	[ECUC_Cdd_00048]
CddPduRLowerLayerPduRef	1	[ECUC_Cdd_00047]

No Included Containers

⌋

[ECUC_Cdd_00052] Definition of EcucEnumerationParamDef CddPduRApiType ⌈

Parameter Name	CddPduRApiType	
Parent Container	CddPduRLowerLayerRxPdu, CddPduRLowerLayerTxPdu, CddPduRUpperLayerRxPdu, CddPduRUpperLayerTxPdu, CddSoAdUpperLayerRxPdu, CddSoAdUpperLayerTxPdu	
Upstream mapping defined	–	
Description	This parameter configures the type of the CDD interface (IF/TP)	
Multiplicity	0..1	
Type	EcucEnumerationParamDef	
Range	IF	IF (Interface)
	TP	TP (TransportProtocol)
Pre-Compile only	true	

⌋

[ECUC_Cdd_00048] Definition of EcucIntegerParamDef CddPduRLowerLayerHandleId ⌈

Parameter Name	CddPduRLowerLayerHandleId	
Parent Container	CddPduRLowerLayerRxPdu	
Upstream mapping defined	–	
Description	ECU wide unique, symbolic handle for the Pdu.	
Multiplicity	0..1	
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)	
Range	0 .. 65535	
Default value	–	
With Auto	true	
Pre-Compile only	true	

⌋

[ECUC_Cdd_00047] Definition of EcucReferenceDef CddPduRLowerLayerPduRef ⌈

Parameter Name	CddPduRLowerLayerPduRef
Parent Container	CddPduRLowerLayerRxPdu
Upstream mapping defined	–

▽

△

Description	Reference to the "global" Pdu structure to allow harmonization of handle IDs in the COM-Stack.
Multiplicity	1
Type	Reference to Pdu
Pre-Compile only	true

⌋

3.5.2 L-Sdu Router

Complex Drivers, which interact with the L-Sdu Router module, define their contribution using the CddLSduRUpperLayerContribution and CddLSduRLowerLayerContribution.

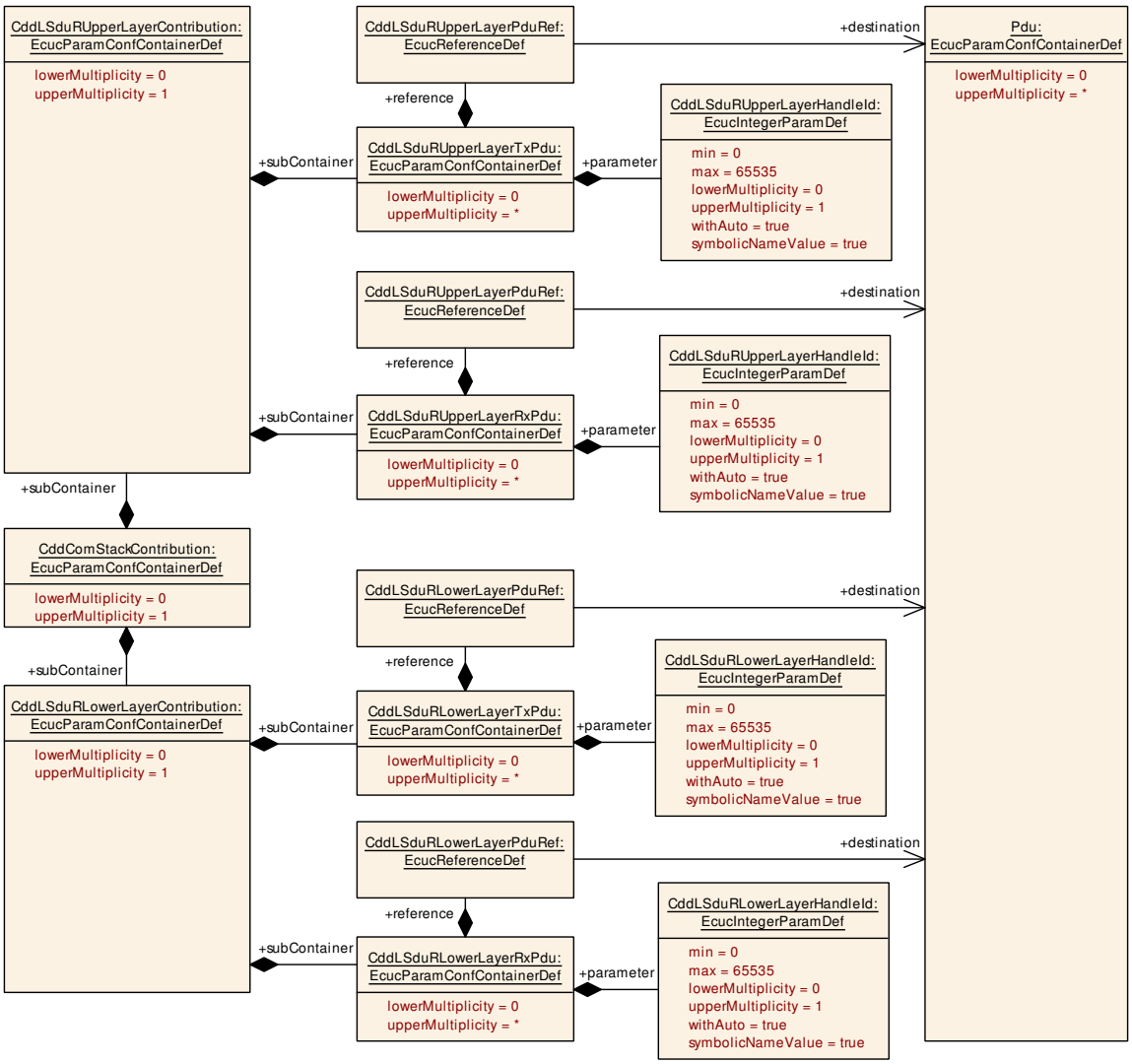

Figure 3.24: IEEE1722Tp Stream contribution

[ECUC_Cdd_00092] Definition of EcucParamConfContainerDef CddLSduRUpperLayerContribution

Status: DRAFT

⌈

Container Name	CddLSduRUpperLayerContribution
Parent Container	CddComStackContribution
Upstream mapping defined	–
Description	Parameters that are necessary for the configuration of a Complex Driver that serves as the UpperLayer of the L-Sdu Router module. **Tags:** atp.Status=draft
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

No Included Parameters

Included Containers		
Container Name	Multiplicity	Description
CddLSduRUpperLayerRxPdu	0..*	This container specifies Rx PDUs that are exchanged between an upper layer CDD and the L-Sdu Router. **Tags:** atp.Status=draft
CddLSduRUpperLayerTxPdu	0..*	This container specifies Tx PDUs that are exchanged between an upper layer CDD and the L-Sdu Router. **Tags:** atp.Status=draft

⌋

[ECUC_Cdd_00086] Definition of EcucParamConfContainerDef CddLSduRUpperLayerTxPdu

Status: DRAFT

⌈

Container Name	CddLSduRUpperLayerTxPdu
Parent Container	CddLSduRUpperLayerContribution
Upstream mapping defined	–
Description	This container specifies Tx PDUs that are exchanged between an upper layer CDD and the L-Sdu Router. **Tags:** atp.Status=draft
Multiplicity	0..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
CddLSduRUpperLayerHandleId	0..1	[ECUC_Cdd_00090]
CddLSduRUpperLayerPduRef	1	[ECUC_Cdd_00089]

No Included Containers

⌋

[ECUC_Cdd_00090] Definition of EcucIntegerParamDef CddLSduRUpperLayerHandleId

Status: DRAFT

⌈

Parameter Name	CddLSduRUpperLayerHandleId
Parent Container	CddLSduRUpperLayerTxPdu
Upstream mapping defined	–
Description	ECU wide unique, symbolic handle for the Pdu. **Tags:** atp.Status=draft
Multiplicity	0..1
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)
Range	0 .. 65535
Default value	–
With Auto	true
Pre-Compile only	true

⌋

[ECUC_Cdd_00089] Definition of EcucReferenceDef CddLSduRUpperLayerPduRef

Status: DRAFT

⌈

Parameter Name	CddLSduRUpperLayerPduRef
Parent Container	CddLSduRUpperLayerTxPdu
Upstream mapping defined	–
Description	Reference to the "global" Pdu structure to allow harmonization of handle IDs in the COM-Stack. **Tags:** atp.Status=draft
Multiplicity	1
Type	Reference to Pdu
Pre-Compile only	true

⌋

[ECUC_Cdd_00087] Definition of EcucParamConfContainerDef CddLSduRUpperLayerRxPdu

Status: DRAFT

⌈

Container Name	CddLSduRUpperLayerRxPdu
Parent Container	CddLSduRUpperLayerContribution
Upstream mapping defined	–

▽

△

Description	This container specifies Rx PDUs that are exchanged between an upper layer CDD and the L-Sdu Router. **Tags:** atp.Status=draft
Multiplicity	0..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
CddLSduRUpperLayerHandleId	0..1	[ECUC_Cdd_00091]
CddLSduRUpperLayerPduRef	1	[ECUC_Cdd_00088]

No Included Containers

⌋

[ECUC_Cdd_00091] Definition of EcucIntegerParamDef CddLSduRUpperLayerHandleId

Status: DRAFT

⌈

Parameter Name	CddLSduRUpperLayerHandleId
Parent Container	CddLSduRUpperLayerRxPdu
Upstream mapping defined	–
Description	ECU wide unique, symbolic handle for the Pdu. **Tags:** atp.Status=draft
Multiplicity	0..1
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)
Range	0 .. 65535
Default value	–
With Auto	true
Pre-Compile only	true

⌋

[ECUC_Cdd_00088] Definition of EcucReferenceDef CddLSduRUpperLayerPduRef

Status: DRAFT

⌈

Parameter Name	CddLSduRUpperLayerPduRef
Parent Container	CddLSduRUpperLayerRxPdu
Upstream mapping defined	–
Description	Reference to the "global" Pdu structure to allow harmonization of handle IDs in the COM-Stack. **Tags:** atp.Status=draft

▽

△

Multiplicity	1
Type	Reference to Pdu
Pre-Compile only	true

⌋

[ECUC_Cdd_00099] Definition of EcucParamConfContainerDef CddLSduRLowerLayerContribution

Status: DRAFT

⌈

Container Name	CddLSduRLowerLayerContribution
Parent Container	CddComStackContribution
Upstream mapping defined	–
Description	Parameters that are necessary for the configuration of a Complex Driver that serves as the LowerLayer of the L-Sdu Router module. **Tags:** atp.Status=draft
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

No Included Parameters

Included Containers		
Container Name	**Multiplicity**	**Description**
CddLSduRLowerLayerRxPdu	0..*	This container specifies Rx PDUs that are exchanged between a lower layer CDD and the L-Sdu Router. **Tags:** atp.Status=draft
CddLSduRLowerLayerTxPdu	0..*	This container specifies Tx PDUs that are exchanged between a lower layer CDD and the L-Sdu Router. **Tags:** atp.Status=draft

⌋

[ECUC_Cdd_00093] Definition of EcucParamConfContainerDef CddLSduRLowerLayerTxPdu

Status: DRAFT

⌈

Container Name	CddLSduRLowerLayerTxPdu
Parent Container	CddLSduRLowerLayerContribution
Upstream mapping defined	–
Description	This container specifies Tx PDUs that are exchanged between a lower layer CDD and the L-Sdu Router. **Tags:** atp.Status=draft
Multiplicity	0..*

▽

△

Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
CddLSduRLowerLayerHandleId	0..1	[ECUC_Cdd_00097]
CddLSduRLowerLayerPduRef	1	[ECUC_Cdd_00096]

No Included Containers

⌋

[ECUC_Cdd_00097] Definition of EcucIntegerParamDef CddLSduRLowerLayerHandleId

Status: DRAFT

⌈

Parameter Name	CddLSduRLowerLayerHandleId
Parent Container	CddLSduRLowerLayerTxPdu
Upstream mapping defined	–
Description	ECU wide unique, symbolic handle for the Pdu. **Tags:** atp.Status=draft
Multiplicity	0..1
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)
Range	0 .. 65535
Default value	–
With Auto	true
Pre-Compile only	true

⌋

[ECUC_Cdd_00096] Definition of EcucReferenceDef CddLSduRLowerLayerPduRef

Status: DRAFT

⌈

Parameter Name	CddLSduRLowerLayerPduRef
Parent Container	CddLSduRLowerLayerTxPdu
Upstream mapping defined	–
Description	Reference to the "global" Pdu structure to allow harmonization of handle IDs in the COM-Stack. **Tags:** atp.Status=draft
Multiplicity	1
Type	Reference to Pdu
Pre-Compile only	true

⌋

[ECUC_Cdd_00094] Definition of EcucParamConfContainerDef CddLSduRLowerLayerRxPdu

Status: DRAFT

⌈

Container Name	CddLSduRLowerLayerRxPdu
Parent Container	CddLSduRLowerLayerContribution
Upstream mapping defined	–
Description	This container specifies Rx PDUs that are exchanged between a lower layer CDD and the L-Sdu Router. **Tags:** atp.Status=draft
Multiplicity	0..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
CddLSduRLowerLayerHandleId	0..1	[ECUC_Cdd_00098]
CddLSduRLowerLayerPduRef	1	[ECUC_Cdd_00095]

No Included Containers

⌋

[ECUC_Cdd_00098] Definition of EcucIntegerParamDef CddLSduRLowerLayerHandleId

Status: DRAFT

⌈

Parameter Name	CddLSduRLowerLayerHandleId	
Parent Container	CddLSduRLowerLayerRxPdu	
Upstream mapping defined	–	
Description	ECU wide unique, symbolic handle for the Pdu. **Tags:** atp.Status=draft	
Multiplicity	0..1	
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)	
Range	0 .. 65535	
Default value	–	
With Auto	true	
Pre-Compile only	true	

⌋

[ECUC_Cdd_00095] Definition of EcucReferenceDef CddLSduRLowerLayerPduRef

Status: DRAFT

⌈

Parameter Name	CddLSduRLowerLayerPduRef
Parent Container	CddLSduRLowerLayerRxPdu
Upstream mapping defined	–
Description	Reference to the "global" Pdu structure to allow harmonization of handle IDs in the COM-Stack. **Tags:** atp.Status=draft
Multiplicity	1
Type	Reference to Pdu
Pre-Compile only	true

⌋

3.5.3 COM Interface modules

A Complex Driver is not allowed to access the COM Stack modules FrDrv, CanDrv and LinDrv. For these modules there is no more than one user. Therefore the lower layer of the COM Stack Bus Interface modules (FrIf, LinIf, CanIf) is not regarded in the CDD module. Upper layer Complex Drivers are allowed to access the interface of these modules. Equal to the `PduRContribution` the `CddComIfUpperLayerContribution` of the Complex Driver implies a reference to the global Pdu and the definition of a HandleId. Figure 3.22 shows the CDD contribution in the configuration model.

Note that the optional presence of the *TxPdu* and *RxPdu* does not influence the existence of the respective APIs in the *Cdd*.

[ECUC_Cdd_00018] Definition of EcucParamConfContainerDef CddComIfUpperLayerContribution ⌈

Container Name	CddComIfUpperLayerContribution
Parent Container	CddComStackContribution
Upstream mapping defined	–
Description	Parameters that are necessary for the configuration of a Complex Driver that serves as the UpperLayer of the Com Interface module.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

No Included Parameters

Included Containers		
Container Name	**Multiplicity**	**Description**
CddComIfUpperLayerRxPdu	0..*	This container specifies Rx PDUs that are exchanged between the CDD and the standardized BSW module.
CddComIfUpperLayerTxPdu	0..*	This container specifies Tx PDUs that are exchanged between the CDD and the standardized BSW module.

⌋

[ECUC_Cdd_00019] Definition of EcucParamConfContainerDef CddComIfUpperLayerTxPdu ⌈

Container Name	CddComIfUpperLayerTxPdu
Parent Container	CddComIfUpperLayerContribution
Upstream mapping defined	–
Description	This container specifies Tx PDUs that are exchanged between the CDD and the standardized BSW module.
Multiplicity	0..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
CddComIfHandleId	0..1	[ECUC_Cdd_00021]
CddComIfPduRef	1	[ECUC_Cdd_00020]

No Included Containers

⌋

[ECUC_Cdd_00021] Definition of EcucIntegerParamDef CddComIfHandleId ⌈

Parameter Name	CddComIfHandleId
Parent Container	CddComIfUpperLayerTxPdu
Upstream mapping defined	–
Description	ECU wide unique, symbolic handle for the Pdu.
Multiplicity	0..1
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)
Range	0 .. 65535
Default value	–
With Auto	true
Pre-Compile only	true

⌋

[ECUC_Cdd_00020] Definition of EcucReferenceDef CddComIfPduRef ⌈

Parameter Name	CddComIfPduRef
Parent Container	CddComIfUpperLayerTxPdu
Upstream mapping defined	–

▽

△

Description	Reference to the "global" Pdu structure to allow harmonization of handle IDs in the COM-Stack.
Multiplicity	1
Type	Reference to Pdu
Pre-Compile only	true

⌋

[ECUC_Cdd_00049] Definition of EcucParamConfContainerDef CddComIfUpperLayerRxPdu ⌈

Container Name	CddComIfUpperLayerRxPdu
Parent Container	CddComIfUpperLayerContribution
Upstream mapping defined	–
Description	This container specifies Rx PDUs that are exchanged between the CDD and the standardized BSW module.
Multiplicity	0..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
CddComIfHandleId	0..1	[ECUC_Cdd_00051]
CddComIfPduRef	1	[ECUC_Cdd_00050]

No Included Containers

⌋

[ECUC_Cdd_00051] Definition of EcucIntegerParamDef CddComIfHandleId ⌈

Parameter Name	CddComIfHandleId	
Parent Container	CddComIfUpperLayerRxPdu	
Upstream mapping defined	–	
Description	ECU wide unique, symbolic handle for the Pdu.	
Multiplicity	0..1	
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)	
Range	0 .. 65535	
Default value	–	
With Auto	true	
Pre-Compile only	true	

⌋

[ECUC_Cdd_00050] Definition of EcucReferenceDef CddComIfPduRef ⌈

Parameter Name	CddComIfPduRef
Parent Container	CddComIfUpperLayerRxPdu
Upstream mapping defined	–
Description	Reference to the "global" Pdu structure to allow harmonization of handle IDs in the COM-Stack.
Multiplicity	1
Type	Reference to Pdu
Pre-Compile only	true

⌋

3.5.4 Communication Manager

Complex Drivers are allowed to access the Communication Manager on the upper layer. The contribution of the lower layer Complex Driver implies for each channel a reference to to the unique handle to identify one certain network handle in the ComM configuration.

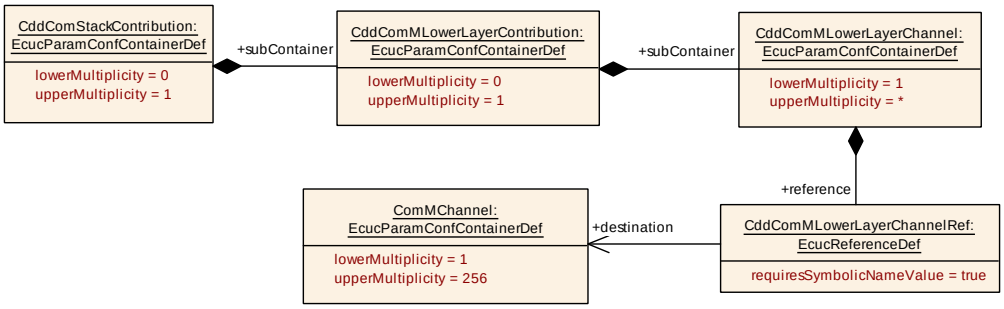

Figure 3.25: ComM lower layer contribution

[ECUC_Cdd_00030] Definition of EcucParamConfContainerDef CddComMLower LayerContribution ⌈

Container Name	CddComMLowerLayerContribution
Parent Container	CddComStackContribution
Upstream mapping defined	–
Description	Parameters that are necessary for the configuration of a Complex Driver that serves as the LowerLayer of the Communication Manager module.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

No Included Parameters

Included Containers		
Container Name	Multiplicity	Description
CddComMLowerLayerChannel	1..*	This container contains the network specific parameters.

⌋

[ECUC_Cdd_00031] Definition of EcucParamConfContainerDef CddComMLowerLayerChannel ⌈

Container Name	CddComMLowerLayerChannel
Parent Container	CddComMLowerLayerContribution
Upstream mapping defined	–
Description	This container contains the network specific parameters.
Multiplicity	1..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
CddComMLowerLayerChannelRef	1	[ECUC_Cdd_00032]

No Included Containers

⌋

[ECUC_Cdd_00032] Definition of EcucReferenceDef CddComMLowerLayerChannelRef ⌈

Parameter Name	CddComMLowerLayerChannelRef
Parent Container	CddComMLowerLayerChannel
Upstream mapping defined	–
Description	Unique handle to identify one certain network. Reference to one of the network handles configured for the ComM.
Multiplicity	1
Type	Symbolic name reference to ComMChannel
Pre-Compile only	true

⌋

3.5.5 Generic Network Management

Complex Drivers are allowed to access the GenericNm module on the upper layer. The contribution of the lower layer Complex Driver implies in each `NmChannel` configuration a reference to the respective NM channel handle.

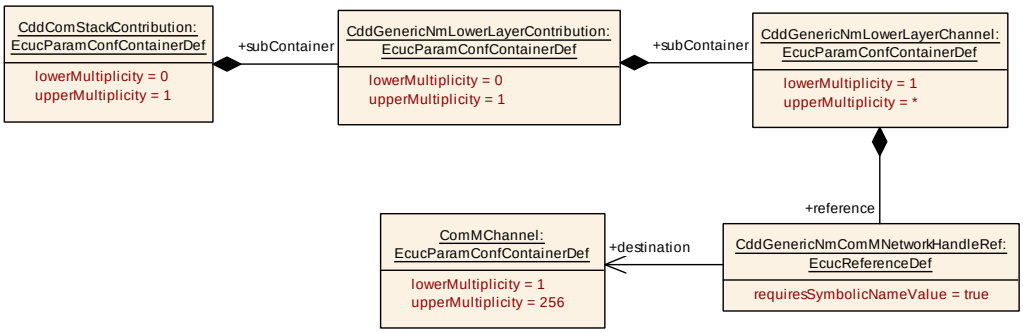

Figure 3.26: GenericNm lower layer contribution

[ECUC_Cdd_00033] Definition of EcucParamConfContainerDef CddGenericNmLowerLayerContribution ⌈

Container Name	CddGenericNmLowerLayerContribution
Parent Container	CddComStackContribution
Upstream mapping defined	–
Description	Parameters that are necessary for the configuration of a Complex Driver that serves as the LowerLayer of the Generic NM module.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

No Included Parameters

Included Containers		
Container Name	**Multiplicity**	**Description**
CddGenericNmLowerLayerChannel	1..*	NM Channel specific configuration parameters.

⌋

[ECUC_Cdd_00034] Definition of EcucParamConfContainerDef CddGenericNmLowerLayerChannel ⌈

Container Name	CddGenericNmLowerLayerChannel
Parent Container	CddGenericNmLowerLayerContribution
Upstream mapping defined	–
Description	NM Channel specific configuration parameters.
Multiplicity	1..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
CddGenericNmComMNetworkHandleRef	1	[ECUC_Cdd_00035]

No Included Containers

⌋

[ECUC_Cdd_00035] Definition of EcucReferenceDef CddGenericNmComMNetworkHandleRef ⌈

Parameter Name	CddGenericNmComMNetworkHandleRef
Parent Container	CddGenericNmLowerLayerChannel
Upstream mapping defined	–
Description	This reference points to the unique channel defined by the ComMChannel and provides access to the unique channel index value in ComMChannelId.
Multiplicity	1
Type	Symbolic name reference to ComMChannel
Pre-Compile only	true

⌋

3.5.6 Socket Adaptor

Complex Drivers are allowed to access the SoAd module on the upper layer. The Pdus that are exchanged between the CDD and the SoAd shall be configured. The contribution of the Complex Driver implies a reference to the global Pdu and the definition of a HandleId. Figure 3.27 shows the CDD contribution in the configuration model.

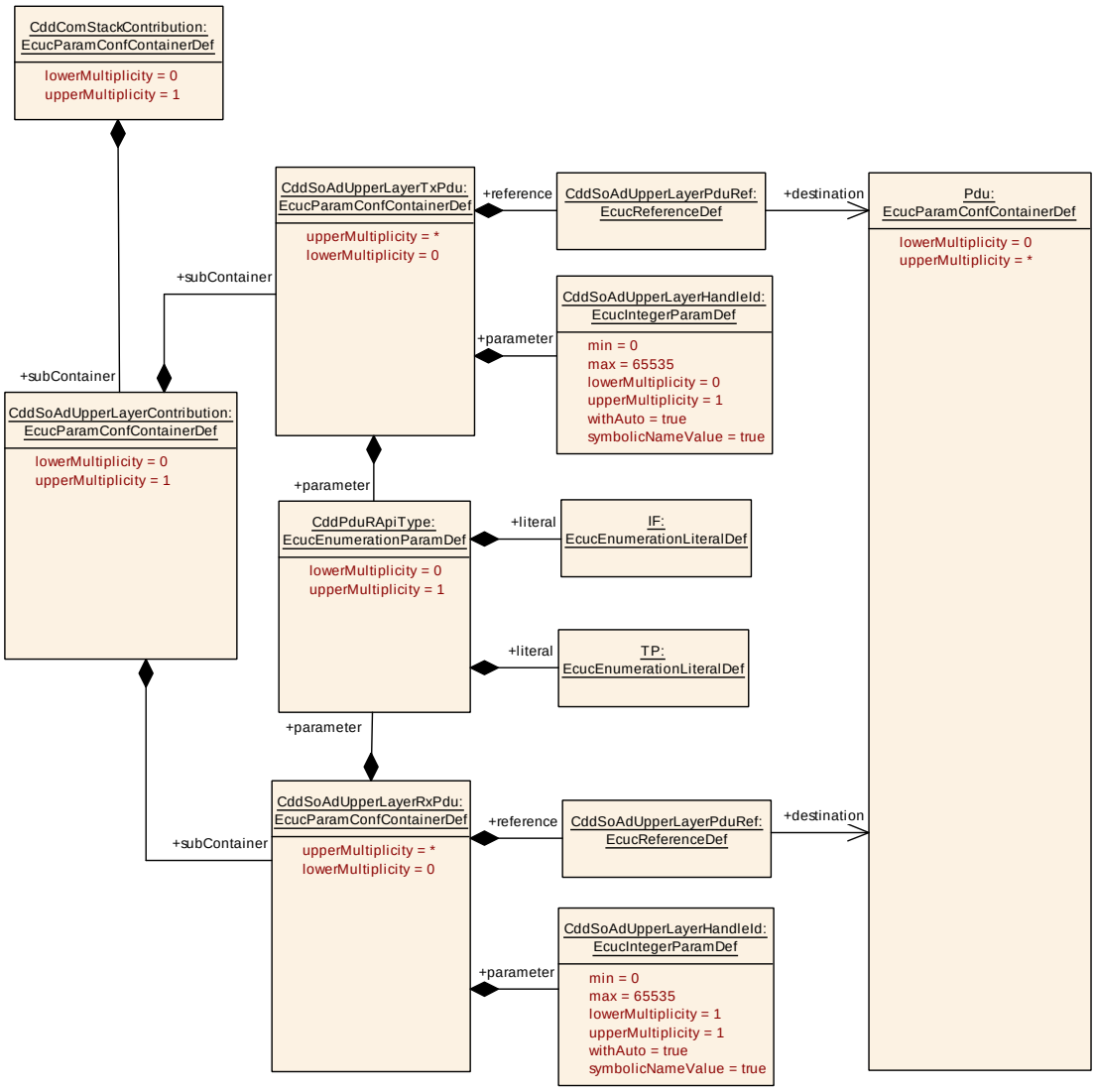

Figure 3.27: SoAd contribution

[ECUC_Cdd_00060] Definition of EcucParamConfContainerDef CddSoAdUpperLayerContribution ⌈

Container Name	CddSoAdUpperLayerContribution
Parent Container	CddComStackContribution
Upstream mapping defined	–
Description	Parameters that are necessary for the configuration of a Complex Driver that serves as the UpperLayer of the SoAd module.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

No Included Parameters

Included Containers		
Container Name	**Multiplicity**	**Description**
CddSoAdUpperLayerRxPdu	0..*	This container specifies Rx PDUs that are exchanged between the CDD and the standardized BSW module.
CddSoAdUpperLayerTxPdu	0..*	This container specifies Tx PDUs that are exchanged between the CDD and the standardized BSW module.

⌋

[ECUC_Cdd_00061] Definition of EcucParamConfContainerDef CddSoAdUpperLayerTxPdu ⌈

Container Name	CddSoAdUpperLayerTxPdu
Parent Container	CddSoAdUpperLayerContribution
Upstream mapping defined	–
Description	This container specifies Tx PDUs that are exchanged between the CDD and the standardized BSW module.
Multiplicity	0..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
CddPduRApiType	0..1	[ECUC_Cdd_00052]
CddSoAdUpperLayerHandleId	0..1	[ECUC_Cdd_00064]
CddSoAdUpperLayerPduRef	1	[ECUC_Cdd_00063]

No Included Containers

⌋

For parameter table [ECUC_Cdd_00052] CddPduRApiType, see definition below container CddPduRLowerLayerRxPdu.

[ECUC_Cdd_00064] Definition of EcucIntegerParamDef CddSoAdUpperLayerHandleId ⌈

Parameter Name	CddSoAdUpperLayerHandleId	
Parent Container	CddSoAdUpperLayerTxPdu	
Upstream mapping defined	–	
Description	ECU wide unique, symbolic handle for the Pdu.	
Multiplicity	0..1	
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)	
Range	0 .. 65535	
Default value	–	
With Auto	true	
Pre-Compile only	true	

⌋

[ECUC_Cdd_00063] Definition of EcucReferenceDef CddSoAdUpperLayerPduRef ⌈

Parameter Name	CddSoAdUpperLayerPduRef
Parent Container	CddSoAdUpperLayerTxPdu
Upstream mapping defined	–
Description	Reference to the "global" Pdu structure to allow harmonization of handle IDs in the COM-Stack.
Multiplicity	1
Type	Reference to Pdu
Pre-Compile only	true

⌋

[ECUC_Cdd_00062] Definition of EcucParamConfContainerDef CddSoAdUpperLayerRxPdu ⌈

Container Name	CddSoAdUpperLayerRxPdu
Parent Container	CddSoAdUpperLayerContribution
Upstream mapping defined	–
Description	This container specifies Rx PDUs that are exchanged between the CDD and the standardized BSW module.
Multiplicity	0..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
CddPduRApiType	0..1	[ECUC_Cdd_00052]
CddSoAdUpperLayerHandleId	1	[ECUC_Cdd_00066]
CddSoAdUpperLayerPduRef	1	[ECUC_Cdd_00065]

No Included Containers

⌋

For parameter table [ECUC_Cdd_00052] CddPduRApiType, see definition below container CddPduRLowerLayerRxPdu.

[ECUC_Cdd_00066] Definition of EcucIntegerParamDef CddSoAdUpperLayerHandleId ⌈

Parameter Name	CddSoAdUpperLayerHandleId	
Parent Container	CddSoAdUpperLayerRxPdu	
Upstream mapping defined	–	
Description	ECU wide unique, symbolic handle for the Pdu.	
Multiplicity	1	
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)	
Range	0 .. 65535	

▽

△

Default value	–
With Auto	true
Pre-Compile only	true

⌋

[ECUC_Cdd_00065] Definition of EcucReferenceDef CddSoAdUpperLayerPduRef ⌈

Parameter Name	CddSoAdUpperLayerPduRef
Parent Container	CddSoAdUpperLayerRxPdu
Upstream mapping defined	–
Description	Reference to the "global" Pdu structure to allow harmonization of handle IDs in the COM-Stack.
Multiplicity	1
Type	Reference to Pdu
Pre-Compile only	true

⌋

3.5.7 J1939Rm

The J1939Rm provides a CDD interface with several callout functions. To be able to generate a header file for a CDD that can in turn be included in J1939Rm to make the callout prototypes available a J1939Rm CDD contribution is available.

[ECUC_Cdd_00079] Definition of EcucParamConfContainerDef CddJ1939Rm Contribution ⌈

Container Name	CddJ1939RmContribution
Parent Container	CddComStackContribution
Upstream mapping defined	–
Description	Contribution of J1939Rm module
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
CddJ1939RmAckIndication	1	[ECUC_Cdd_00081]
CddJ1939RmRequestIndication	1	[ECUC_Cdd_00080]
CddJ1939RmRequestTimeoutIndication	1	[ECUC_Cdd_00082]

No Included Containers

⌋

[ECUC_Cdd_00081] Definition of EcucBooleanParamDef CddJ1939RmAckIndication ⌈

Parameter Name	CddJ1939RmAckIndication
Parent Container	CddJ1939RmContribution
Upstream mapping defined	–
Description	Defines whether the <User>_AckIndication callback function is implemented.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_Cdd_00080] Definition of EcucBooleanParamDef CddJ1939RmRequestIndication ⌈

Parameter Name	CddJ1939RmRequestIndication
Parent Container	CddJ1939RmContribution
Upstream mapping defined	–
Description	Defines whether the <User>_RequestIndication callback function is implemented.
Multiplicity	1

▽

△

Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_Cdd_00082] Definition of EcucBooleanParamDef CddJ1939RmRequestTimeoutIndication ⌈

Parameter Name	CddJ1939RmRequestTimeoutIndication
Parent Container	CddJ1939RmContribution
Upstream mapping defined	–
Description	Defines whether the <User>_RequestTimeoutIndication callback function is implemented.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[TPS_ECUC_06093] J1939Rm callback functions ⌈The prototypes of the configured J1939Rm callback functions shall be exported to a header file named <apiServicePrefix>_J1939Rm.h.⌋

3.5.8 Global Time Synchronization

Complex Drivers, which implement Timebase Providers for Global Time Synchronization, are allowed to access the StbM to manage the synchronized time-bases.

Figure 3.28 shows the CDD contribution in the configuration model.

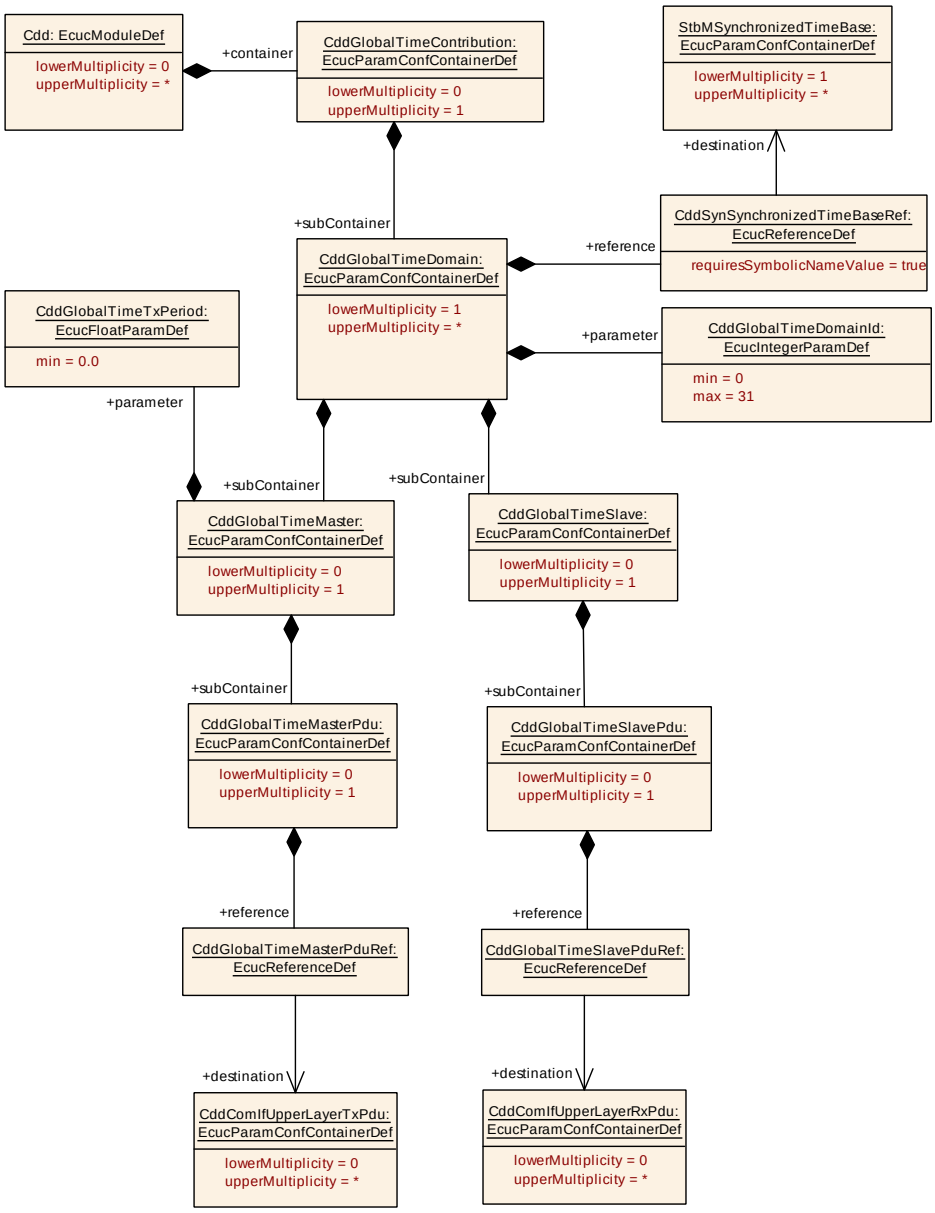

Figure 3.28: Global time contribution

[ECUC_Cdd_00068] Definition of EcucParamConfContainerDef CddGlobalTime Contribution ⌈

Container Name	CddGlobalTimeContribution
Parent Container	Cdd
Upstream mapping defined	–
Description	Contribution of Global Time modules.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

No Included Parameters

Included Containers		
Container Name	Multiplicity	Description
CddGlobalTimeDomain	1..*	This represents the existence of a CDD global time domain. The CddGlobalTimeContribution can administrate several global time domains at the same time that in itself form a hierarchy of domains and sub-domains.

⌋

[ECUC_Cdd_00069] Definition of EcucParamConfContainerDef CddGlobalTimeDomain ⌈

Container Name	CddGlobalTimeDomain
Parent Container	CddGlobalTimeContribution
Upstream mapping defined	–
Description	This represents the existence of a CDD global time domain. The CddGlobalTimeContribution can administrate several global time domains at the same time that in itself form a hierarchy of domains and sub-domains.
Multiplicity	1..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
CddGlobalTimeDomainId	1	[ECUC_Cdd_00071]
CddSynSynchronizedTimeBaseRef	1	[ECUC_Cdd_00070]

Included Containers		
Container Name	Multiplicity	Description
CddGlobalTimeMaster	0..1	Configuration of the global time master. Each global time domain is required to have exactly one global time master. This master may or may not exist on the configured ECU.
CddGlobalTimeSlave	0..1	Configuration of a global time slave. Each global time domain is required to have at least one time slave. The configured ECU may or may not represent a time slave.

⌋

[ECUC_Cdd_00071] Definition of EcucIntegerParamDef CddGlobalTimeDomainId ⌈

Parameter Name	CddGlobalTimeDomainId	
Parent Container	CddGlobalTimeDomain	
Upstream mapping defined	–	
Description	The global time domain ID.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 31	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_Cdd_00070] Definition of EcucReferenceDef CddSynSynchronizedTimeBaseRef ⌈

Parameter Name	CddSynSynchronizedTimeBaseRef
Parent Container	CddGlobalTimeDomain
Upstream mapping defined	–
Description	Mandatory reference to the required synchronized time-base.
Multiplicity	1
Type	Symbolic name reference to StbMSynchronizedTimeBase
Pre-Compile only	true

⌋

[ECUC_Cdd_00072] Definition of EcucParamConfContainerDef CddGlobalTimeMaster ⌈

Container Name	CddGlobalTimeMaster
Parent Container	CddGlobalTimeDomain
Upstream mapping defined	–
Description	Configuration of the global time master. Each global time domain is required to have exactly one global time master. This master may or may not exist on the configured ECU.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
CddGlobalTimeTxPeriod	1	[ECUC_Cdd_00074]

Included Containers		
Container Name	**Multiplicity**	**Description**
CddGlobalTimeMasterPdu	0..1	This container encloses the configuration of the PDU that is supposed to contain the global time information. Please note that the configuration of CddGlobalTimeMasterPdu is optional and shall only be used for Complex Drivers that are using Pdus for carrying timeSync information.

⌋

[ECUC_Cdd_00074] Definition of EcucFloatParamDef CddGlobalTimeTxPeriod ⌈

Parameter Name	CddGlobalTimeTxPeriod	
Parent Container	CddGlobalTimeMaster	
Upstream mapping defined	[1, CP_TPS_SystemTemplate]	
Description	This represents configuration of the TX period. Unit: seconds	
Multiplicity	1	
Type	EcucFloatParamDef	
Range	[0 .. INF]	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_Cdd_00073] Definition of EcucParamConfContainerDef CddGlobalTime Slave ⌈

Container Name	CddGlobalTimeSlave
Parent Container	CddGlobalTimeDomain
Upstream mapping defined	–
Description	Configuration of a global time slave. Each global time domain is required to have at least one time slave. The configured ECU may or may not represent a time slave.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

No Included Parameters

Included Containers		
Container Name	**Multiplicity**	**Description**
CddGlobalTimeSlavePdu	0..1	This container encloses the configuration of the PDU that is supposed to contain the global time information. Please note that the configuration of CddGlobalTimeSlavePdu is optional and shall only be used for Complex Drivers that are using Pdus for carrying timeSync information.

⌋

[ECUC_Cdd_00075] Definition of EcucParamConfContainerDef CddGlobalTime MasterPdu ⌈

Container Name	CddGlobalTimeMasterPdu
Parent Container	CddGlobalTimeMaster
Upstream mapping defined	–

▽

△

Description	This container encloses the configuration of the PDU that is supposed to contain the global time information. Please note that the configuration of CddGlobalTimeMasterPdu is optional and shall only be used for Complex Drivers that are using Pdus for carrying timeSync information.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
CddGlobalTimeMasterPduRef	1	[ECUC_Cdd_00076]

No Included Containers

⌋

[ECUC_Cdd_00076] Definition of EcucReferenceDef CddGlobalTimeMasterPduRef ⌈

Parameter Name	CddGlobalTimeMasterPduRef
Parent Container	CddGlobalTimeMasterPdu
Upstream mapping defined	–
Description	This represents the reference to the Pdu taken to transmit the global time information. The global time master of a global time domain is the sender of this Pdu.
Multiplicity	1
Type	Reference to CddComIfUpperLayerTxPdu
Pre-Compile only	true

⌋

[ECUC_Cdd_00077] Definition of EcucParamConfContainerDef CddGlobalTimeSlavePdu ⌈

Container Name	CddGlobalTimeSlavePdu
Parent Container	CddGlobalTimeSlave
Upstream mapping defined	–
Description	This container encloses the configuration of the PDU that is supposed to contain the global time information. Please note that the configuration of CddGlobalTimeSlavePdu is optional and shall only be used for Complex Drivers that are using Pdus for carrying timeSync information.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
CddGlobalTimeSlavePduRef	1	[ECUC_Cdd_00078]

No Included Containers

⌋

[ECUC_Cdd_00078] Definition of EcucReferenceDef CddGlobalTimeSlavePduRef ⌈

Parameter Name	CddGlobalTimeSlavePduRef
Parent Container	CddGlobalTimeSlavePdu
Upstream mapping defined	–
Description	This represents the reference to the Pdu taken to transmit the global time information. All the time slaves are supposed to receive the Pdu.
Multiplicity	1
Type	Reference to CddComIfUpperLayerRxPdu
Pre-Compile only	true

⌋

3.6 EcuM configuration to initialize post-build capable BSW Modules

The `EcuMDriverInitItem` contains `EcuMModuleRef` references to configurations (`EcucModuleConfigurationValues`) of module instances which shall be initialized by EcuM.

`EcucModuleConfigurationValues` may contain `VariationPoint`s. In order to initialize a post-build capable BSW module the reference in the `VariationPoint` to the `PostBuildVariantCriterion` with the right `PostBuildVariantCriterionValue` shall be used (see section 2.4.7).

Which `PredefinedVariant`s exist is defined by `EcucPostBuildVariants` as described in section 3.3.3.

3.7 Optional reporting of Production Errors and Extended Production Errors

The reporting of Production errors from any BSW Module to the Dem is configurable (see figure 2.17 for an example). The respective `EcucReferenceDef`s with the attribute `requiresSymbolicNameValue` set to `true` from the reporting module to the `DemEventParameter` are optional.

[TPS_ECUC_02143] Optional configuration of Production Error and Extended Production Error reporting ⌈The configuration of Production Error and Extended Production Error reporting is optional for the reporting BSW module. Due to further functional requirements in the reporting BSW Module it may still be required to detect the Production Error or Extended Production Error and behave accordingly, even when the reporting to the Dem is not configured.⌋

Another possibility is to configure and report the Production Error or Extended Production Error to the Dem and then filter inside the Dem configuration the behavior for this `DemEventParameter` such that it will not have an effect.

3.8 Converting time parameters of main functions to ticks

Typically the time related parameters in AUTOSAR are given as float values. Nevertheless for some parameters the unit [ticks] is required. The advantage of having ticks in the ECU configuration is that the final value is already known before the code generator is called. Otherwise it depends on the implementer of the code generator what final value is calculated.

[TPS_ECUC_08010] Ticks in the Ecuc Parameter Values ⌈An error shall be generated if the generated number of ticks with the current main cycle does not match the desired timing.⌋

3.9 Clock Tree Configuration

In the standardized ECU Configuration Parameter Definition, only HW independent parameters can be specified. Since the clock tree is highly HW dependent the MCU clock reference point has been introduced which allows an abstract description of clock properties independent of the hardware.

Thus the details of the clock tree configuration shall be hardware/vendor specific additions to the MCU Driver Configuration added by the implementor of the MCU Driver. This means, that other drivers (possibly vendor specific), such as CAN Driver, need a mechanism to derive the correct settings for their timing registers, since they do not know the actual hardware specific parameters.

The MCU module defines a container McuClockReferencePoint (multiplicity 1..*). In this container a parameter McuClockReferencePointFrequency (type float, in Hz) is provided.

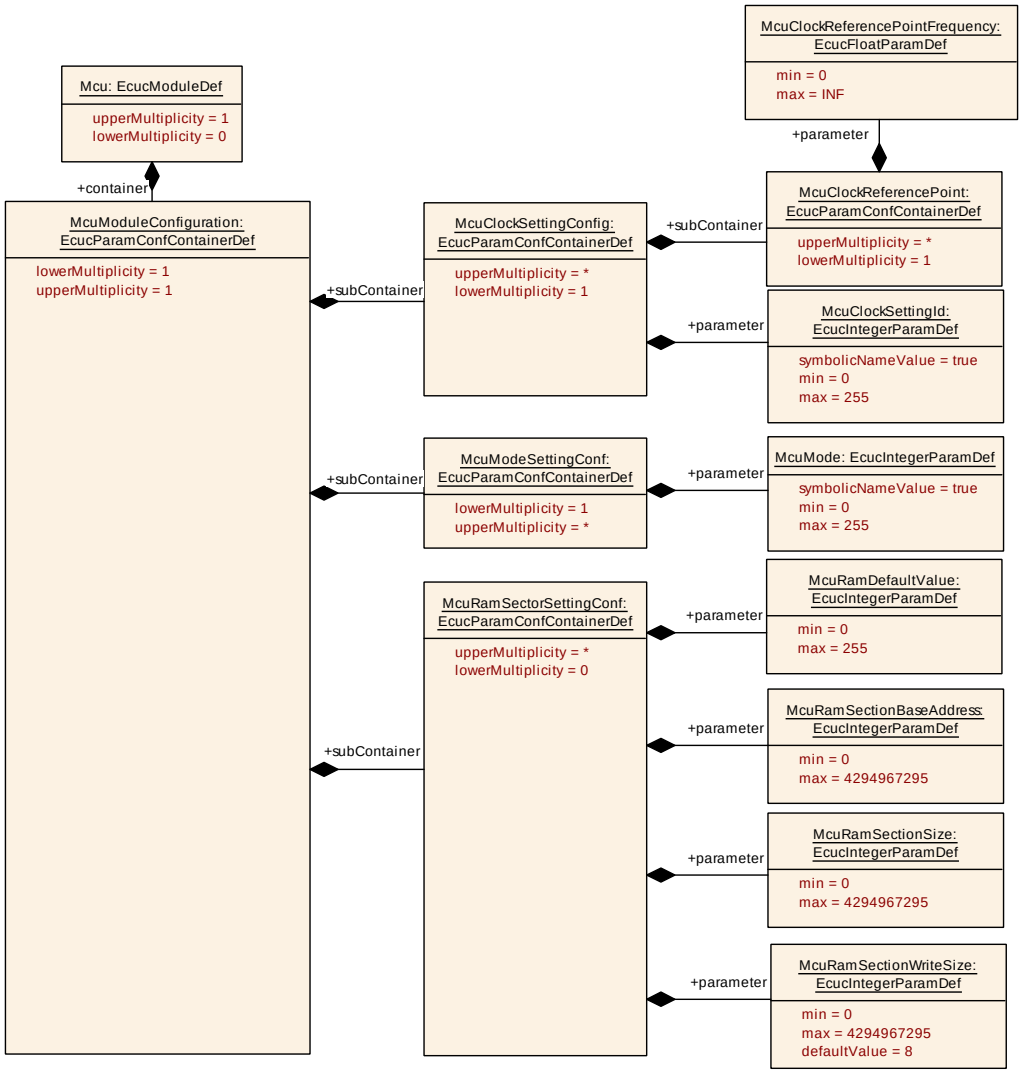

Figure 3.29: MCU Setting

[ECUC_Mcu_00174] Definition of EcucParamConfContainerDef McuClockReferencePoint

Container Name	McuClockReferencePoint
Parent Container	McuClockSettingConfig
Upstream mapping defined	–
Description	This container defines a reference point in the Mcu Clock tree. It defines the frequency which then can be used by other modules as an input value. Lower multiplicity is 1, as even in the simplest case (only one frequency is used), there is one frequency to be defined.
Multiplicity	1..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
McuClockReferencePointFrequency	1	[ECUC_Mcu_00175]

No Included Containers

[ECUC_Mcu_00175] Definition of EcucFloatParamDef McuClockReferencePoint Frequency

Parameter Name	McuClockReferencePointFrequency	
Parent Container	McuClockReferencePoint	
Upstream mapping defined	–	
Description	This is the frequency for the specific instance of the McuClockReferencePoint container. It shall be given in Hz.	
Multiplicity	1	
Type	EcucFloatParamDef	
Range	[0 .. INF]	
Default value	–	
Post-Build Value	Loadable	Selectable

The ECU integrator and/or MCU configuration/generation tool need to derive from those required output frequencies - together with other parameters such as input clock frequency - how its internal settings for pre-scalers, muxes, etc. need to be configured.

The users of clock frequencies (e.g. CanDrv, LinDrv, PWM) define in their configuration a reference to the container `McuClockReferencePoint` that allows them to select which input clock they choose. In that container the modules generator will find the frequency to use as input frequency (value of parameter `McuClockReferencePointFrequency`). The users of clock frequencies might need to adjust the clock further by setting local pre-scalers and dividers.

The configuration editor for the peripheral module (i.e. CanDrv configuration editor) can support the integrator by only allowing a selection of those clock reference points that can be connected physically to that peripheral.

The design guideline is that all settings until the MCU clock reference point are under the responsibility of the MCU Driver (see figure 3.30). Further adjustments on the clock frequency are under the responsibility of the specific user peripheral's driver.

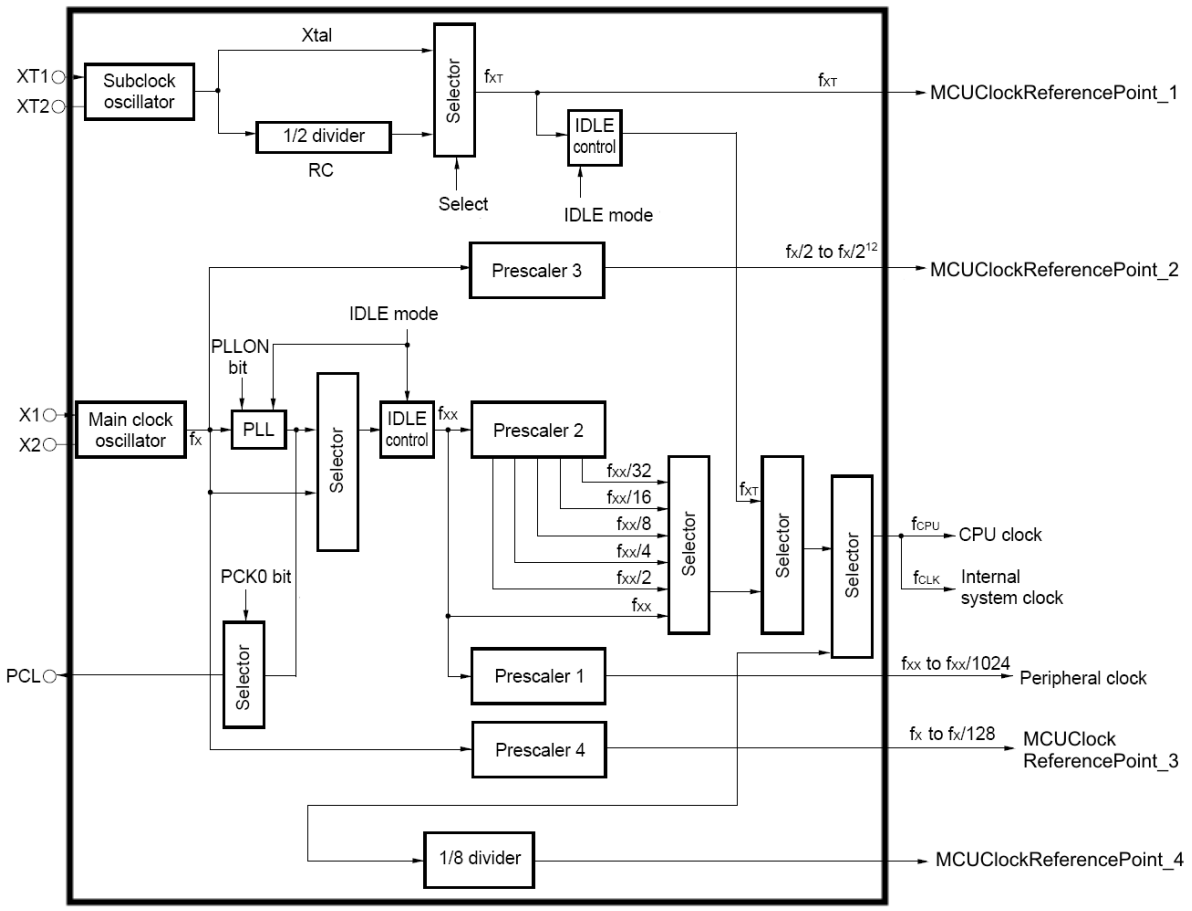

Figure 3.30: Clocktree example

4 Rules to follow in different configuration activities

This chapter defines rules relevant for the relation between standardized module definitions and vendor specific module definitions, rules for building the base `ECU Configuration Values` and rules for configuration editors. The generation of the base `ECU Configuration Values` as a part of the ECU configuration process is explained in the AUTOSAR Methodology ([2] Chapter 2.7.3 *"Prepare ECU Configuration"* and [2] Chapter 3.6.1.3 *"Generate Base ECU Configuration"*).

4.1 Deriving vendor specific module definitions from standardized module definitions

The basic relationship between the `Vendor Specific Module Definition` (abbreviated with VSMD in this chapter) and `Standardized Module Definition` (abbreviated StMD in this chapter) is depicted in figure 4.1.

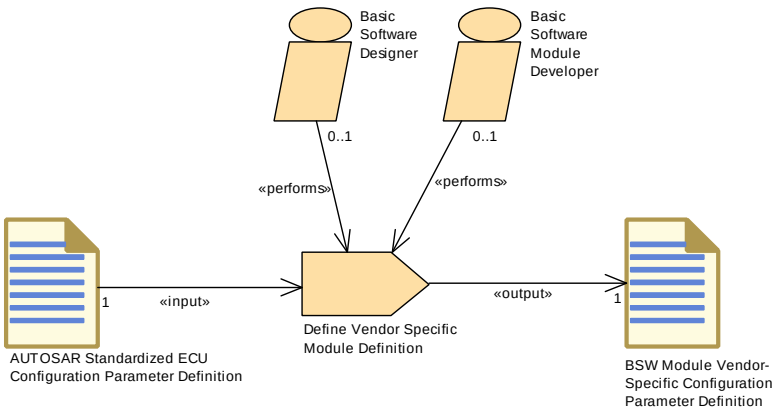

Figure 4.1: Generating Vendor Specific Module Definitions (per module)

Please note that also a pure VSMD which has no counterpart in the StMD is allowed to exist. Vendor specific parameters/containers/references with no relationship to StMD may also be available in a VSMD. Figure 4.2 shows an example with pure vendor specific containers and references (marked with red color).

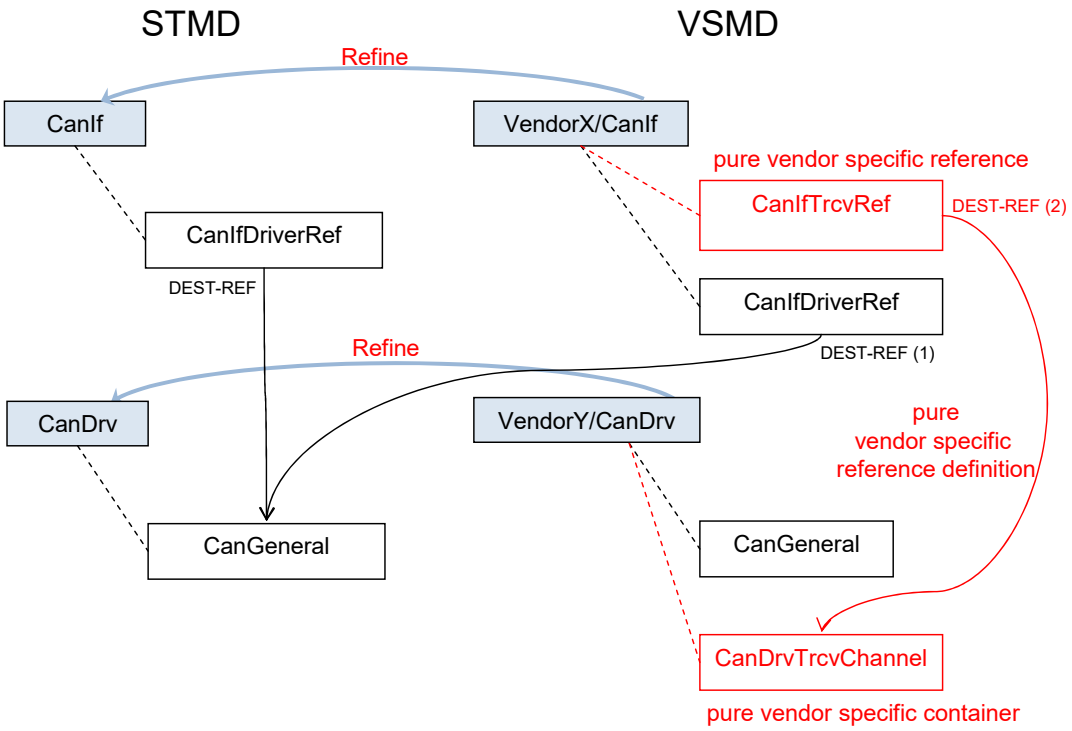

Figure 4.2: Relation between STMD and VSMD

In Listing 4.1 the StMD of the two modules of figure 4.2 is defined.

```
<AR-PACKAGE>
  <SHORT-NAME>AUTOSAR</SHORT-NAME>
  <AR-PACKAGES>
    <AR-PACKAGE>
      <SHORT-NAME>EcucDefs</SHORT-NAME>
      <ELEMENTS>
        <ECUC-MODULE-DEF>
          <SHORT-NAME>CanIf</SHORT-NAME>
          <CONTAINERS>
            <ECUC-PARAM-CONF-CONTAINER-DEF>
              <SHORT-NAME>CanIfDriver</SHORT-NAME>
              <REFERENCES>
                <ECUC-REFERENCE-DEF>
                  <SHORT-NAME>CanIfDriverRef</SHORT-NAME>
                  <DESTINATION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF">/
                      AUTOSAR/EcucDefs/CanDrv/CanGeneral</DESTINATION-REF>
                </ECUC-REFERENCE-DEF>
              </REFERENCES>
            </ECUC-PARAM-CONF-CONTAINER-DEF>
          </CONTAINERS>
        </ECUC-MODULE-DEF>
        <ECUC-MODULE-DEF>
          <SHORT-NAME>CanDrv</SHORT-NAME>
          <CONTAINERS>
            <ECUC-PARAM-CONF-CONTAINER-DEF>
              <SHORT-NAME>CanGeneral</SHORT-NAME>
            </ECUC-PARAM-CONF-CONTAINER-DEF>
          </CONTAINERS>
```

```
        </ECUC-MODULE-DEF>
      </ELEMENTS>
    </AR-PACKAGE>
  </AR-PACKAGES>
</AR-PACKAGE>
```

Listing 4.1: `Standardized Module Definition` (StMD) Example

In Listing 4.2 the VSMD of a `CanDrv` implementation is shown.

```
<AR-PACKAGE>
  <SHORT-NAME>VendorY</SHORT-NAME>
  <ELEMENTS>
    <ECUC-MODULE-DEF>
      <SHORT-NAME>CanDrv</SHORT-NAME>
      <REFINED-MODULE-DEF-REF DEST="ECUC-MODULE-DEF">/AUTOSAR/EcucDefs/
          CanDrv</REFINED-MODULE-DEF-REF>
      <CONTAINERS>
        <ECUC-PARAM-CONF-CONTAINER-DEF>
          <SHORT-NAME>CanGeneral</SHORT-NAME>
        </ECUC-PARAM-CONF-CONTAINER-DEF>
        <ECUC-PARAM-CONF-CONTAINER-DEF>
          <SHORT-NAME>CanTrcvChannel</SHORT-NAME>
        </ECUC-PARAM-CONF-CONTAINER-DEF>
      </CONTAINERS>
    </ECUC-MODULE-DEF>
  </ELEMENTS>
</AR-PACKAGE>
```

Listing 4.2: `Vendor Specific Module Definition` (VSMD) for the previous StMD

In Listing 4.3 the VSMD of a `CanIf` implementation is shown. The implicitly refined reference `CanIfDriverRef` still has the DESTINATION-REF in the VSMD pointing to the standardized AUTOSAR short-name path.

Additionally, the pure vendor specific reference `CanIfDrvTrcvRef` has been introduced, which points to the vendor specific container `CanTrcvChannel` using the DESTINATION-REF with a fully qualified vendor specific short-name path.

```
<AR-PACKAGE>
  <SHORT-NAME>VendorX</SHORT-NAME>
  <ELEMENTS>
    <ECUC-MODULE-DEF>
      <SHORT-NAME>CanIf</SHORT-NAME>
      <REFINED-MODULE-DEF-REF DEST="ECUC-MODULE-DEF">/AUTOSAR/EcucDefs/
          CanIf</REFINED-MODULE-DEF-REF>
      <CONTAINERS>
        <ECUC-PARAM-CONF-CONTAINER-DEF>
          <SHORT-NAME>CanIfDriver</SHORT-NAME>
          <REFERENCES>
            <ECUC-REFERENCE-DEF>
              <SHORT-NAME>CanIfDriverRef</SHORT-NAME>
              <DESTINATION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF">/
                  AUTOSAR/EcucDefs/CanDrv/CanGeneral</DESTINATION-REF>
            </ECUC-REFERENCE-DEF>
            <ECUC-REFERENCE-DEF>
```

```
            <SHORT-NAME>CanIfDrvTrcvRef</SHORT-NAME>
            <DESTINATION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF">/
               VendorY/CanDrv/CanTrcvChannel</DESTINATION-REF>
          </ECUC-REFERENCE-DEF>
        </REFERENCES>
      </ECUC-PARAM-CONF-CONTAINER-DEF>
    </CONTAINERS>
  </ECUC-MODULE-DEF>
 </ELEMENTS>
</AR-PACKAGE>
```

Listing 4.3: VSMD with additional vendor specific container `CanIfDrvTrcvRef`

[TPS_ECUC_06038] Rules to validate a BSW module implementation ⌈The following rules shall be checked by tools that validate whether a SW module implementation conforms to its AUTOSAR specification.⌋

- **[TPS_ECUC_01001] `lowerMultiplicity` and `upperMultiplicity` of modules in the VSMD** ⌈The `lowerMultiplicity` of the module in the VSMD shall be equal or bigger to what is defined in the StMD. The `upperMultiplicity` of that module shall be equal or less to what is defined in the StMD. StMD lowerMult $\leq$ VSMD lowerMult $\leq$ VSMD upperMult $\leq$ StMD upperMult.⌋

- **[TPS_ECUC_06001] `shortName` of a VSMD module** ⌈The `shortName` of a VSMD module shall be the same as the `shortName` of the StMD.⌋

- **[TPS_ECUC_06049] Restriction of `supportedConfigVariant`s in the VSMD** ⌈The supported `EcucModuleDef.supportedConfigVariant` shall be restricted in the VSMD to the actually supported configuration variants of this implementation. This can be a subset of the `EcucModuleDef.supportedConfigVariant` in the StMD.⌋

- **[TPS_ECUC_06003] Package structure of the VSMD** ⌈The package structure of the VSMD has to be different than "/AUTOSAR/EcucDefs/" so that it is possible to distinguish the standardized from the vendor specific module definitions.⌋

  Note: Listings 4.4, 4.5 and 4.6 show the difference between the VSMD and StMD. The package structure of the vendor specific CanIf module definition begins with "/VendorX/CanIf" (Listing 4.5) and the package structure of the vendor specific CanDrv module definition begins with "/VendorY/Can" (Listing 4.6).

- **[TPS_ECUC_06015] DESTINATION-REF in the VSMD** ⌈The DESTINATION-REF in the VSMD shall point to the standardized AUTOSAR short-name path (e.g. /AUTOSAR/EcucDefs/Can/CanController) if the reference definition has an STMD counterpart. In this case the vendor specific short-name path (e.g. /VendorX/Can) shall not be used.⌋

Note: Listing 4.5 shows a DESTINATION-REF from the CanIf module provided from VendorX to the CanDrv module provided by VendorY. The DESTINATION-REF content is not changed from "/AUTOSAR/EcucDefs/..." in the VSMD.

- **[TPS_ECUC_06046] Vendor specific reference definition with no counterpart in the STMD** ⌈A pure vendor specific reference definition (which has no counterpart in the STMD) can refer either
  - to a standardized container (has a counterpart in the STMD) or
  - to a vendor specific container.

  In either case it is possible to use the fully qualified vendor specific short-name path for the DESTINATION-REF. Only for the first option (reference to standardized container) it is alternatively possible to use the standardized AUTOSAR short-name path.⌋

```
<AR-PACKAGE>
  <SHORT-NAME>AUTOSAR</SHORT-NAME>
  <AR-PACKAGES>
    <AR-PACKAGE>
      <SHORT-NAME>EcucDefs</SHORT-NAME>
      <ELEMENTS>
        <ECUC-MODULE-DEF>
          <SHORT-NAME>CanIf</SHORT-NAME>
          <CONTAINERS>
            <ECUC-PARAM-CONF-CONTAINER-DEF>
              <SHORT-NAME>CanIfDriverConfig</SHORT-NAME>
              <REFERENCES>
                <!--Reference Definition:CanIfDriverRef-->
                <ECUC-REFERENCE-DEF>
                  <SHORT-NAME>CanIfDriverRef</SHORT-NAME>
                  <DESTINATION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF">/
                      AUTOSAR/EcucDefs/Can/CanGeneral</DESTINATION-REF>
                </ECUC-REFERENCE-DEF>
              </REFERENCES>
            </ECUC-PARAM-CONF-CONTAINER-DEF>
          </CONTAINERS>
        </ECUC-MODULE-DEF>
        <ECUC-MODULE-DEF>
          <SHORT-NAME>Can</SHORT-NAME>
          <CONTAINERS>
            <ECUC-PARAM-CONF-CONTAINER-DEF>
              <SHORT-NAME>CanGeneral</SHORT-NAME>
              <PARAMETERS>
                <!-- ... -->
              </PARAMETERS>
            </ECUC-PARAM-CONF-CONTAINER-DEF>
          </CONTAINERS>
        </ECUC-MODULE-DEF>
```

Listing 4.4: CanIf and CanDrv AUTOSAR standardized XML

```
<AR-PACKAGE>
  <SHORT-NAME>VendorX</SHORT-NAME>
```

```
  <ELEMENTS>
    <ECUC-MODULE-DEF>
      <SHORT-NAME>CanIf</SHORT-NAME>
      <REFINED-MODULE-DEF-REF DEST="ECUC-MODULE-DEF">/AUTOSAR/EcucDefs/
          CanIf</REFINED-MODULE-DEF-REF>
      <CONTAINERS>
        <ECUC-PARAM-CONF-CONTAINER-DEF>
          <SHORT-NAME>CanIfDriverConfig</SHORT-NAME>
          <REFERENCES>
            <!--Reference Definition:CanIfDriverRef-->
            <ECUC-REFERENCE-DEF>
              <SHORT-NAME>CanIfDriverRef</SHORT-NAME>
              <DESTINATION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF">/
                  AUTOSAR/EcucDefs/Can/CanGeneral</DESTINATION-REF>
            </ECUC-REFERENCE-DEF>
          </REFERENCES>
        </ECUC-PARAM-CONF-CONTAINER-DEF>
      </CONTAINERS>
    </ECUC-MODULE-DEF>
  </ELEMENTS>
</AR-PACKAGE>
```

Listing 4.5: CanIf VendorX XML

```
<AR-PACKAGE>
  <SHORT-NAME>VendorY</SHORT-NAME>
  <ELEMENTS>
    <ECUC-MODULE-DEF>
      <SHORT-NAME>Can</SHORT-NAME>
      <REFINED-MODULE-DEF-REF DEST="ECUC-MODULE-DEF">/AUTOSAR/EcucDefs/Can<
          /REFINED-MODULE-DEF-REF>
      <CONTAINERS>
        <ECUC-PARAM-CONF-CONTAINER-DEF>
          <SHORT-NAME>CanGeneral</SHORT-NAME>
          <PARAMETERS>
            <!-- ... -->
          </PARAMETERS>
        </ECUC-PARAM-CONF-CONTAINER-DEF>
      </CONTAINERS>
    </ECUC-MODULE-DEF>
  </ELEMENTS>
</AR-PACKAGE>
```

Listing 4.6: CanDrv VendorY XML

For all EcucContainerDefs and EcucParameterDefs and EcucAbstractReferenceDefs defined within the EcucModuleDef in the StMD, it holds:

- **[TPS_ECUC_06007] Elements defined in the StMD shall be present in the VSMD** ⌈Elements defined in the StMD shall be present in the VSMD and shall not be omitted, even if the upperMultiplicity of an element in the VSMD is set to 0.⌋

- **[TPS_ECUC_06008] lowerMultiplicity and upperMultiplicity of elements in the VSMD** ⌈The lowerMultiplicity of an element in the VSMD

shall be bigger or equal and the upperMultiplicity shall be equal or less than in the StMD:

StMD lowerMult ≤ VSMD lowerMult ≤ VSMD upperMult ≤ StMD upperMult.⌋

- **[TPS_ECUC_06109] The value of the EcucModuleDef.postBuildLoadableSupport attribute in the VSMD in case it is not defined in the StMD** ⌈If the postBuildLoadableSupport attribute for an EcucModuleDef is not defined in the StMD, the corresponding VSMD shall set it to either false or true.⌋

- **[TPS_ECUC_06110] The value of the EcucModuleDef.postBuildLoadableSupport attribute in the VSMD in case it is set to false in the StMD** ⌈If the postBuildLoadableSupport attribute for an EcucModuleDef is set to false, the corresponding VSMD shall also set it to false.⌋

  This means that if the value of the postBuildLoadableSupport attribute for one BSW module is set to false in the StMD, this BSW module does not support an update of the configuration at post-build time.

- **[TPS_ECUC_06111] The value of the EcucModuleDef.postBuildLoadableSupport attribute in the VSMD in case it is set to true in the StMD** ⌈If the postBuildLoadableSupport attribute for an EcucModuleDef is set to true, the corresponding VSMD shall set it to either false or true.⌋

  This means that if the value of the postBuildLoadableSupport attribute for one BSW module is set to true in the StMD, this BSW module supports an update of the configuration at post-build time which may or may not be used.

- **[TPS_ECUC_06112] The value of the EcucContainerDef.postBuildLoadableMultiplicity attribute in the VSMD in case it is set to false in the StMD** ⌈If the EcucModuleDef.postBuildLoadableSupport is set to true and the postBuildLoadableMultiplicity attribute of an EcucContainerDef in this EcucModuleDef in the StMD is set to false, the corresponding VSMD may set the postBuildLoadableMultiplicity attribute to either false or true (if [constr_5506] is fulfilled).⌋

- **[TPS_ECUC_06113] The value of the EcucContainerDef.postBuildLoadableMultiplicity in the VSMD in case it is set to true in the StMD** ⌈If the EcucModuleDef.postBuildLoadableSupport is set to true and the postBuildLoadableMultiplicity attribute of an EcucContainerDef in this EcucModuleDef in the StMD is set to true, the corresponding VSMD shall also set the postBuildLoadableMultiplicity to true.⌋

- **[TPS_ECUC_06114] The value of the EcucParameterDef.postBuildLoadableValue and the EcucAbstractReferenceDef.postBuildLoadableValue attributes in the VSMD in case they are set to false in**

the StMD ⌈If the EcucModuleDef.postBuildLoadableSupport is set to true and the postBuildLoadableValue for an EcucParameterDef or an EcucAbstractReferenceDef in this EcucModuleDef in the StMD is set to false, the corresponding VSMD may set it to either false or true.⌋

- **[TPS_ECUC_06115] The value of the EcucParameterDef.postBuildLoadableMultiplicity and the EcucAbstractReferenceDef.postBuildLoadableMultiplicity attributes in the VSMD in case they are set to false in the StMD** ⌈If the EcucModuleDef.postBuildLoadableSupport is set to true and the postBuildLoadableMultiplicity for an EcucParameterDef or an EcucAbstractReferenceDef in this EcucModuleDef in the StMD is set to false, the corresponding VSMD may set it to either false or true.⌋

- **[TPS_ECUC_06116] The value of the EcucParameterDef.postBuildLoadableValue and the EcucAbstractReferenceDef.postBuildLoadableValue attributes in the VSMD in case they are set to true in the StMD** ⌈If the EcucModuleDef.postBuildLoadableSupport is set to true and the postBuildLoadableValue for an EcucParameterDef or an EcucAbstractReferenceDef in this EcucModuleDef in the StMD is set to true, the corresponding VSMD shall also set it to true.⌋

- **[TPS_ECUC_06117] The value of the EcucParameterDef.postBuildLoadableMultiplicity and the EcucAbstractReferenceDef.postBuildLoadableMultiplicity attributes in the VSMD in case they are set to true in the StMD** ⌈If the EcucModuleDef.postBuildLoadableSupport is set to true and the postBuildLoadableMultiplicity for an EcucParameterDef or an EcucAbstractReferenceDef in this EcucModuleDef in the StMD is set to true, the corresponding VSMD shall also set it to true.⌋

- **[TPS_ECUC_08021] The value of the EcucModuleDef.postBuildVariantSupport attribute in the VSMD in case it is not defined in the StMD** ⌈If the postBuildVariantSupport attribute for an EcucModuleDef is not defined in the StMD, the corresponding VSMD can set it to either false or true.⌋

- **[TPS_ECUC_08041] The value of the EcucModuleDef.postBuildVariantSupport attribute in the VSMD in case it set to false in the StMD** ⌈If the postBuildVariantSupport attribute for an EcucModuleDef is set to false, the corresponding VSMD shall also set it to false.⌋

  This means that if the value of the postBuildVariantSupport attribute for one BSW module is set to false in the StMD, this BSW module does not support variation points bound at post-build time.

- **[TPS_ECUC_08042] The value of the EcucModuleDef.postBuildVariantSupport attribute in the VSMD in case it is set to true in the StMD** ⌈If the

postBuildVariantSupport attribute for an EcucModuleDef is set to true, the corresponding VSMD can set it to either false or true.⌋

This means that if the value of the postBuildVariantSupport attribute for one BSW module is set to true in the StMD, this BSW module supports variation points bound at post-build time which may or may not be used.

- **[TPS_ECUC_08026] The value of the EcucContainerDef.postBuildVariantMultiplicity attribute in the VSMD in case it is set to false in the StMD** ⌈If the EcucModuleDef.postBuildVariantSupport is set to true and the postBuildVariantMultiplicity attribute of an EcucContainerDef in this EcucModuleDef in the StMD is set to false, the corresponding VSMD may set it to either false or true (if [constr_5506] is fulfilled).⌋

- **[TPS_ECUC_08027] The value of the EcucContainerDef.postBuildVariantMultiplicity in the VSMD in case it is set to true in the StMD** ⌈If the EcucModuleDef.postBuildVariantSupport is set to true and the postBuildVariantMultiplicity attribute of an EcucContainerDef in this EcucModuleDef in the StMD is set to true, the corresponding VSMD shall also set it to true.⌋

- **[TPS_ECUC_08030] The value of the EcucParameterDef.postBuildVariantValue and the EcucAbstractReferenceDef.postBuildVariantValue attributes in the VSMD in case they are set to false in the StMD** ⌈If the EcucModuleDef.postBuildVariantSupport is set to true and the postBuildVariantValue for an EcucParameterDef or an EcucAbstractReferenceDef in this EcucModuleDef in the StMD is set to false, the corresponding VSMD may set it to either false or true.⌋

- **[TPS_ECUC_08031] The value of the EcucParameterDef.postBuildVariantMultiplicity and the EcucAbstractReferenceDef.postBuildVariantMultiplicity attributes in the VSMD in case they are set to false in the StMD** ⌈If the EcucModuleDef.postBuildVariantSupport is set to true and the postBuildVariantMultiplicity for an EcucParameterDef or an EcucAbstractReferenceDef in this EcucModuleDef in the StMD is set to false, the corresponding VSMD may set it to either false or true.⌋

- **[TPS_ECUC_08032] The value of the EcucParameterDef.postBuildVariantValue and the EcucAbstractReferenceDef.postBuildVariantValue attributes in the VSMD in case they are set to true in the StMD** ⌈If the EcucModuleDef.postBuildVariantSupport is set to true and the postBuildVariantValue for an EcucParameterDef or an EcucAbstractReferenceDef in this EcucModuleDef in the StMD is set to true, the corresponding VSMD shall also set it to true.⌋

- **[TPS_ECUC_08033] The value of the EcucParameterDef.postBuildVariantMultiplicity and the EcucAbstractReferenceDef.postBuildVariantMultiplicity attributes in the VSMD in case they are set to true in the StMD** ⌈If the EcucModuleDef.postBuildVariantSupport is set to true and the postBuildVariantMultiplicity for an EcucParameterDef or an EcucAbstractReferenceDef in this EcucModuleDef in the StMD is set to true, the corresponding VSMD shall also set it to true.⌋

- **[TPS_ECUC_01034] ShortName of elements in the VSMD that are taken over from the StMD** ⌈Elements taken over from the StMD to the VSMD shall use exactly the same shortName, since the short name identifies the element. This holds for container definitions and individual parameters.⌋

- **[TPS_ECUC_01035] UUID of elements in the VSMD that are taken over from the StMD** ⌈Elements taken over from the StMD to the VSMD shall have unique uuid in each ECU Configuration Values. Thus a new uuid might be generated when taking over an element.⌋

- **[TPS_ECUC_01005] Origin attribute of parameters in the VSMD that are taken over from the StMD** ⌈The origin attribute shall not be changed for any parameter taken over from the StMD, even when attributes of the parameter are modified in the VSMD.⌋

- **[TPS_ECUC_01006] DefaultValues of parameters in the VSMD** ⌈The defaultValue attribute may be changed (or added, if missing).⌋

- **[TPS_ECUC_01007] min, max values of parameters in the VSMD** ⌈The min values specified in the VSMD shall be bigger or equal, the max value shall be less or equal than the corresponding value specified in the StMD:
⌋

  StMD minValue $\leq$ VSMD minValue $\leq$ VSMD maxValue $\leq$ StMD maxValue.

- **[TPS_ECUC_06045] min, max values of parameters in the VSMD in case that the min or max value in the StMD is set to infinite** ⌈If the min value equals *-inf* or the max value equals *inf* in the StMD the min/max values in the VSMD shall be replaced with the actually supported min/max values for this implementation.⌋

- **[TPS_ECUC_01009] Calculation of derived parameters in the StMD may change in the VSMD** ⌈For derived parameters defined in the StMD, the values of the calculationFormula and calculationLanguage may change in the VSMD.⌋

- **[TPS_ECUC_01011] Vendor specific choices in EcucChoiceContainerDefs** ⌈Additional vendor specific choices (i.e. aggregated EcucParamConfContainerDefs) may be added to EcucChoiceContainerDefs in the VSMD.⌋

- **[TPS_ECUC_01013] Vendor specific destinations in EcucChoiceReferenceDefs** ⌈Additional vendor specific references may be added for EcucChoiceReferenceDefs in the VSMD.⌋

- **[TPS_ECUC_01014] Addition of vendor specific parameter definitions, container definitions and references** ⌈Additional vendor specific parameter definitions, container definitions and references shall be added to the VSMD according to the alphabetical order.⌋

- **[TPS_ECUC_01015] Origin attribute in vendor specific elements** ⌈The origin attribute of vendor specific additional elements shall contain the name of the vendor that defines the element.⌋

- **[TPS_ECUC_02084] Addition of vendor specific EcucEnumerationLiteralDefs to an EcucEnumerationParamDef from the StMD** ⌈For an EcucEnumerationParamDef from the StMD there can be additional EcucEnumerationLiteralDefs added in the VSMD if the scope of the EcucEnumerationParamDef is local.⌋

- **[TPS_ECUC_05002] Creation of VSMD from the StMD** ⌈Induce VSMD into the StMD in a simplified manner, so that the configuration can be carried out without any disarray.⌋

- **[TPS_ECUC_05003] desc field of parameters in VSMD** ⌈The desc in VSMD can be used to specify detailed information about the respective parameter.⌋

- **[TPS_ECUC_02134] requiresIndex setting in the VSMD** ⌈The requiresIndex setting may be changed in the VSMD.⌋

- **[TPS_ECUC_02137] EcucValidationConditions from the StMD shall be taken over to the VSMD.** ⌈If the StMD defines any ecucValidationConds they shall be taken to the VSMD.⌋

- **[TPS_ECUC_02138] Addition of vendor specific EcucValidationConditions** ⌈Additional ecucValidationConds may be added to the VSMD. Semantically they shall provide more restrictive validation conditions than the ones defined in the StMD.⌋

Figure 4.3 shows an overview about rules, which shall be checked by tools that validate whether a SW module implementation conforms to its AUTOSAR specification. In this example three parameters are defined in the StMD.

The multiplicity in each of these parameter definitions specifies how often a parameter is allowed to occur in the `ECU Configuration Values`. In the VSMD, optional elements (with `lowerMultiplicity` = 0) shall be present, but the `lowerMultiplicity` and `upperMultiplicity` may be set to 0, as it happens with parameter A (1). The `lowerMultiplicity` of parameters in the VSMD shall be bigger or equal than in the StMD (1, 2, 3). The `upperMultiplicity` of parameters in the VSMD shall be equal (2) or less (1, 3) than in the StMD. New vendor specific parameters may also be added in the VSMD (4).

The VSMD defines which parameters are available in which container and what kind of restrictions are to be respected.

[TPS_ECUC_06009] Existence of a parameter in the `ECU Configuration Values` in case the upperMultiplicity of the parameter definition is zero ⌈If the `upperMultiplicity` of a parameter definition in the VSMD is 0, the parameter may be omitted in the `ECU Configuration Values`. If such a parameter exists in the `ECU Configuration Values` it shall be ignored by the tool (5).⌋

[TPS_ECUC_06010] Existence of a parameter in the `ECU Configuration Values` in case the lowerMultiplicity of the parameter definition is bigger than zero ⌈If the `lowerMultiplicity` of a parameter definition is bigger than 0, the parameter shall exist in the `ECU Configuration Values` (6).⌋

[TPS_ECUC_06011] Missing parameters in the `ECU Configuration Values` ⌈Missing parameters shall be detected by tools (8).⌋

[TPS_ECUC_06012] Parameters without parameter definitions in the `ECU Configuration Values` ⌈If Parameters without parameter definitions exist this shall be reported latest during code generation (9).⌋

[TPS_ECUC_06013] Number of parameters in the `ECU Configuration Values` ⌈The number of parameters in the `ECU Configuration Values` shall not exceed the `upperMultiplicity` of the parameter definition in the VSMD (7).⌋

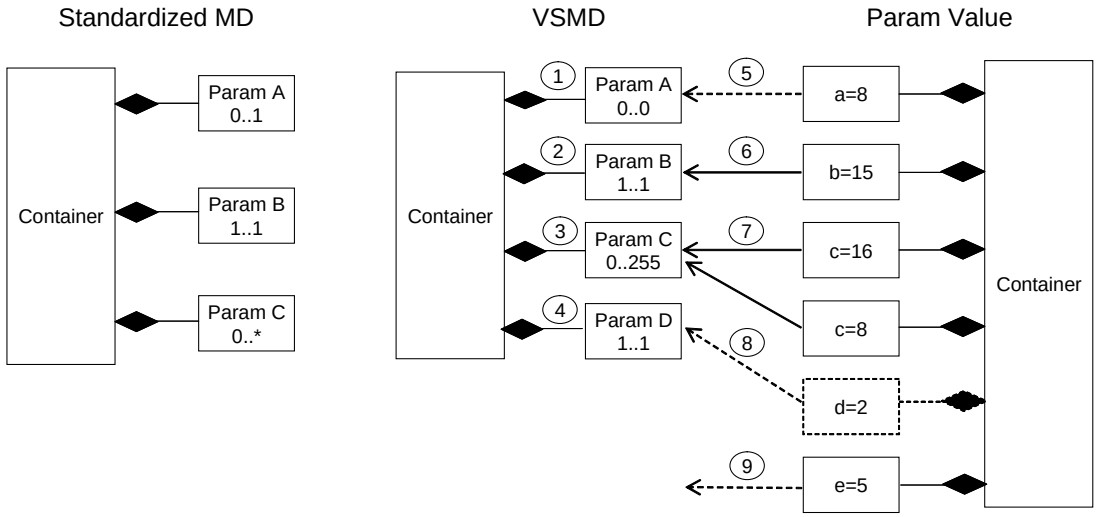

Figure 4.3: Relation between standardized module definition and vendor specific module definition

Example 4.1 depicts the usage of VSMD in case of parameter definition.

Example 4.1

```
<ECUC-INTEGER-PARAM-DEF>
  <SHORT-NAME>ClockRate</SHORT-NAME>
  <ORIGIN>AUTOSAR_ECUC</ORIGIN>
</ECUC-INTEGER-PARAM-DEF>
<ECUC-BOOLEAN-PARAM-DEF>
  <SHORT-NAME>VendorExtensionEnabled</SHORT-NAME>
  <ORIGIN>VendorXYZ_v1.3</ORIGIN>
</ECUC-BOOLEAN-PARAM-DEF>
```

Example 4.2 depicts the usage of VSMD in case of parameter description.

Example 4.2

```
<ECUC-NUMERICAL-PARAM-VALUE>
  <DEFINITION-REF DEST="ECUC-INTEGER-PARAM-DEF">/VendorXY/Mcu/
     McuGeneralConfiguration/ClockRate</DEFINITION-REF>
  <VALUE>123</VALUE>
</ECUC-NUMERICAL-PARAM-VALUE>
<ECUC-NUMERICAL-PARAM-VALUE>
  <DEFINITION-REF DEST="ECUC-BOOLEAN-PARAM-DEF">/VendorXY/Mcu/
     McuGeneralConfiguration/VendorExtensionEnabled</DEFINITION-REF>
  <VALUE>1</VALUE>
</ECUC-NUMERICAL-PARAM-VALUE>
```

In case of a CDD module the configuration parameters for the VSMD of CDD do not specify any configuration class. It is up to the implementor of the specific CDD to define the configuration class for all configuration parameters - standardized and vendor specific ones.

[TPS_ECUC_02139] Definition of precompile/linktime/postbuild support for all CDD configuration parameters and references ⌈For the CDD module the standardized configuration parameters do not specify any precompile/linktime/postbuild-loadable/postbuild-selectable support. It is up to the implementor of the specific CDD module to define the precompile/linktime/postbuild-loadable/postbuild-selectable support for all configuration parameters - standardized and vendor specific ones in the VSMD.⌋

4.2 Rules for building the Base ECU configuration

The AUTOSAR Methodology ([2] Chapter 2.7.3 *"Prepare ECU Configuration"* and [2] Chapter 3.6.1.3 *"Generate Base ECU Configuration"*) defines the activity how to generate the base `ECU Configuration Values`.

The following rules apply during generation of the base ECU configuration for a module:

- **[TPS_ECUC_01016] Generation of instances for mandatory definitions** ⌈For mandatory containers, parameters and references (i.e. with `lowerMultiplicity` > 0 in their definition) at least the number of instances defined by the `lowerMultiplicity` shall be generated.⌋

  E.g. the configuration of a CAN controller may contain the configuration of one or more hardware objects, depending on the hardware. The configuration of hardware objects is done in a subcontainer. Since at least one hardware object is always present, one instance of this subcontainer always has to be present and shall be generated together with the enclosing container for the CAN controller.

- **[TPS_ECUC_01017] Generation of instances for optional definitions** ⌈For optional containers, parameters and references (i.e. with `lowerMultiplicity` $= 0$ in their definition), no instances may be generated.⌋

  E.g. the configuration may contain the definition of RX PDUs in a subcontainer. One subcontainer instance is defined for each PDU received. Since there may be no RX PDUs, it is well possible that no container instance needs to be generated.

- **[TPS_ECUC_01018] Generation of instances for container definitions with variable multiplicity** ⌈For containers with variable multiplicity (i.e. `lowerMultiplicity` $<$ `upperMultiplicity`), any number of instances between lower and upper multiplicity may be generated. (additional instances may be added during editing of the `ECU Configuration Values`).⌋

  E.g., continuing the previous example, several instances may be generated if the definition of RX PDUs can be derived from the ECU extract of System description. If the ECU receives several frames on a CAN bus, at least one RX PDU is normally present per received frame.

- **[TPS_ECUC_01019] Setting of the initial values for configuration parameters** ⌈For the setting of the initial values for configuration parameters, the following sources shall be used (in decreasing priority)⌋

  – **[TPS_ECUC_01020] Values fixed by the implementation as defined in the Vendor Specific Pre-configured ECU Configuration Values** ⌈Since the module implementation fixes those configuration parameters, those values shall be included in the base ECU Configuration Values and shall not be changed in later editing.⌋

  – **[TPS_ECUC_01021] Values derived from the ECU extract of the system configuration.** ⌈The ECU extract may define the basis for the ECU Configuration Values, e.g. for COM stack configuration, the system description provides configuration information for bus speed, frame definitions etc, which can be taken over into the ECU Configuration Values. This derivation of the ECU Configuration Values from the ECU extract of the system configuration shall take place according to the mapping rules defined in annex D "Harmonization between Upstream Templates and ECU Configuration" of [1].⌋

    E.g. The signal definitions relevant for the COM stack can be derived from the ECU extract of system configuration. One container instance with all relevant parameter values taken from the system configuration will be generated for each signal.

  – **[TPS_ECUC_01022] Values provided by the implementor in the BSWMD in the Vendor Specific Recommended Configuration Values.** ⌈Implementors may provide configuration settings in the BSWMD provided with their implementation. This allows the implementor to provide the integrator with his hints which values might be most useful for his implementation of the module on a specific ECU.⌋

  – **[TPS_ECUC_01023] Default values provided as part of the parameter definition.** ⌈Since each configuration parameter is defined only once, all instances of the parameter will have the same initial value when the default values is taken as input to the base configuration.⌋

[TPS_ECUC_01024] Generation of parameters without an initial value ⌈If no initial value can be derived from any of these sources for a parameter, the parameter will be generated without an initial value.⌋

[TPS_ECUC_04004] Iterative development of the ECU Configuration Values ⌈If an existing ECU Configuration Values exist and an updated ECU Extract of System Configuration or BSW Module Description is released, the existing ECU Configuration Values shall be

taken into consideration when updating to a new version of ECU Configuration Values, i.e, the Generate Base Ecu Configuration activity shall consist of a merge functionality. This functionality is optional since the first time an ECU Configuration Values is generated, there is no existing ECU Configuration Values.⌋

4.3 Rules for navigating in ECU Configuration Artifacts

The following rules apply for tools that are navigating in ECU Configuration Artifacts:

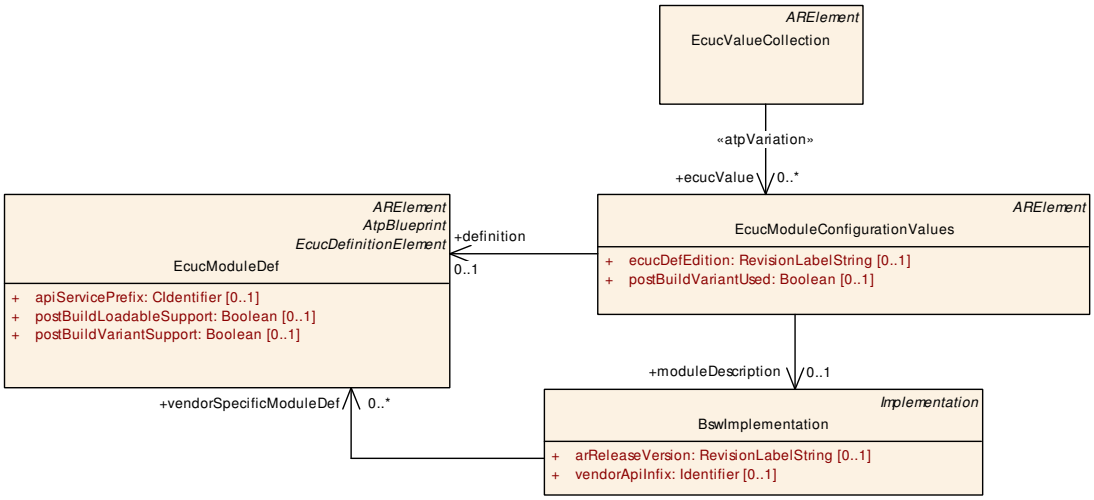

Figure 4.4: ECU Configuration Artifacts

- **[TPS_ECUC_06039] BswImplementation and vendorSpecificModuleDef shall be known by tools** ⌈The tool knows his BswImplementation element and subsequently the vendorSpecificModuleDef.⌋

- **[TPS_ECUC_06040] EcucValueCollection is the input for tools** ⌈The tool shall get the EcucValueCollection as input information. Please note that the input can be provided as multiple files and can be structured in an arbitrary package structure.⌋

- **[TPS_ECUC_06041] Tools shall respect the EcucModuleConfigurationValues elements that are referenced by the EcucValueCollection** ⌈. The tool shall respect only those EcucModuleConfigurationValues elements which are referenced by the EcucValueCollection.⌋

- **[TPS_ECUC_06042] Tools interaction with EcucModuleConfigurationValues** ⌈The tool shall directly interact only with those EcucModuleConfigurationValues elements whose definition reference is equal to the vendorSpecificModuleDef reference from the BswImplementation.⌋

Example: If two CAN drivers from different vendors are to be configured with respective tools each tool can find the `EcucModuleConfigurationValues` to directly interact with using the definition references.

4.4 Post-build Time Consistency

When generating a post-build configuration, it shall be assured that the correct pre-compile and link-time configuration is used.

During initialization, it shall be possible to determine if the pre-compile and link-time configurations matches the post-build configuration (e.g. this can be done by placing a checksum based on the pre-compile and link-time configuration parameters in the pre-compile and link-time configuration and also placing the same checksum in the post-build configuration which are then compared).

Note that pre-compile and link time configuration parameters that are part of the containers that are introduced at post-build time (according to [TPS_ECUC_08002]) shall not be considered for the consistency check.

In addition to this, additional check may be applied in order to assure that the compatible version of the configuration generator is used.

4.5 Impact of ServiceNeeds on ECU Configuration

In the `ECU Extract of the System Configuration` only application Software Components are considered, while RTE and all BSW modules are not taken into account. In contrast, the `ECU Configuration` needs to consider all aspects of the ECU software, therefore means to support the addition of the BSW and RTE need to be provided.

To support this, the `ECU Configuration Description` allows the ECU extract to be extended by adding AUTOSAR Service prototypes and assembly connectors establishing the connections between applications and AUTOSAR service components (see figure 4.5 *EcuTopLevelCompositionPrototype*).

AUTOSAR Services are modules like the NvRam Manager, the Watchdog Manager, the ECU State Manager, etc., which possess the characteristic trait that they interact with application software components using standardized AUTOSAR interfaces.

This topic is a cross cutting concern between ECU Configuration (this document), the Software Component Template [11] and the System Template [1].

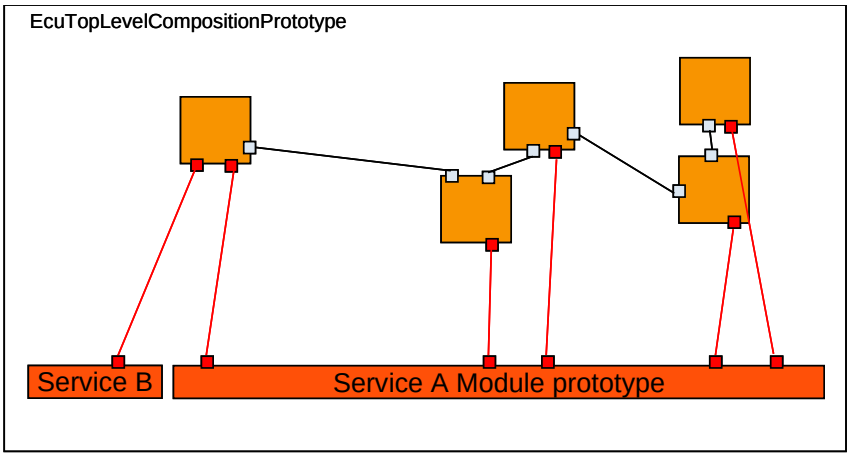

Figure 4.5: Structure of the EcuTopLevelCompositionPrototype introduced in the ECU Configuration

To enable the extension of the existing ECU Extract towards a complete software system in the ECU Configuration, the aggregation of `SwComponentPrototype` and `SwConnector` by `CompositionSwComponentType` is stereotyped ≪`atpSplitable`≫.

This is shown in figure 4.6. Making these aggregations ≪`atpSplitable`≫ allows the addition of AUTOSAR service component prototypes and connector prototypes to the `CompositionSwComponentType` contained in the ECU extract during the ECU integration without changing the artifacts which had been delivered as ECU extract.

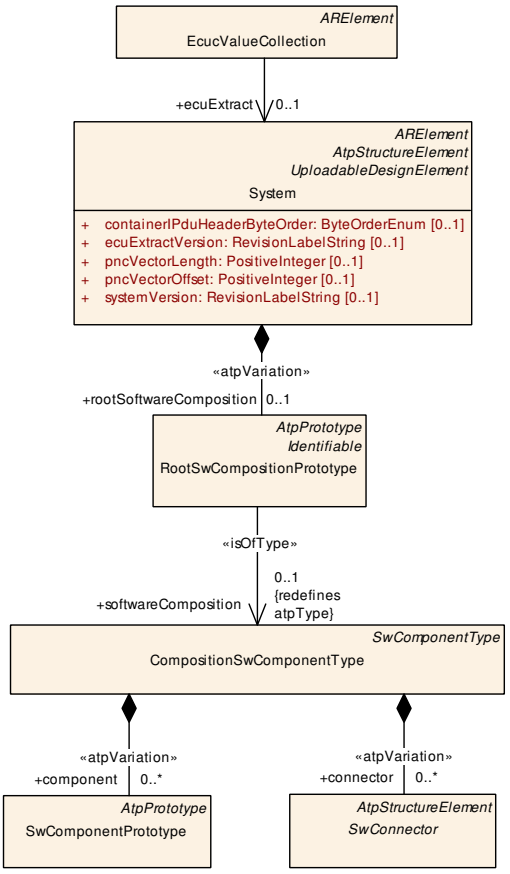

Figure 4.6: Splittable aggregations of `SwComponentPrototype` and `SwConnector`

[TPS_ECUC_02087] Creation of `ServiceSwComponentType`s ⌈When generating the AUTOSAR Service SW-Components the actual *service needs*[1] expressed by the Application SW-Components are collected.

For each AUTOSAR service required, a `ServiceSwComponentType` shall be created complete with an appropriate number of ports to enable the connection of all application component ports expressing the needs for the AUTOSAR service.⌋

[TPS_ECUC_06014] Content of `CompositionSwComponentType` in the ECU Configuration ⌈The `CompositionSwComponentType` in the ECU Configuration shall contain, additionally to prototypes of all application SW-Components running on the ECU as contained in the ECU Extract, `SwComponentPrototype`s for all required AUTOSAR Service modules and `AssemblySwConnector`s for the required connections between the Application SW-Component ports and the AUTOSAR Service module's ports.⌋

[1] The *needs* of the Application SW-Components are defined in the SW-Component description in the `ServiceNeeds` section.

A Reference Material

This chapter contains some relevant reference material for this specification.

A.1 Abbreviations

The content of this appendix chapter is *informative* in nature and shall **not** be considered as *normative* content.

This section describes abbreviations that are specific to the ECU Configuration Specification and that are not part of the official AUTOSAR Glossary [6].

Following abbreviations are mentioned that are specifically used in this specification:

Abbreviation	*meaning*
ECUC	ECU Configuration
ECUC ParamDef	ECU Configuration Parameter Definition
ECUC Value	ECU Configuration Value
StMD	Standardized Module Definition
VSMD	Vendor Specific Module Definition

Table A.1: Abbreviations used in the scope of this Document

A.2 Imposition Times of Constraints

The constraints formulated in this document have different *actual* imposition times which denote the steps in the workflow when the respective constraint has to be imposed.

Please note that the imposition times are intentionally rendered as technical terms such that it is possible to link back from each constraint to the definition of the affected imposition time in Table A.2.

This document has been created to apply for the *AUTOSAR classic platform* and therefore the discussed imposition times also apply exclusively to the *AUTOSAR classic platform*.

Imposition Time	Description	Motivation
IT_EcucDef	ECU Configuration Parameter Definition is complete	This imposition time denotes the step in the workflow where the ECU Configuration Parameter Definition (either a STANDARDIZED_MODULE_DEFINITION or a VENDOR_SPECIFIC_MODULE_DEFINITION) of a Basic Software Module is considered complete. This imposition time is special in the sense that this happens during AUTOSAR Standard development or Stack Vendor Product development.

▽

△

Imposition Time	Description	Motivation
IT_EcucVal	ECU Configuration Values is complete	This imposition time denotes the step in the workflow where the ECU Configuration Values of a Basic Software Module are considered complete such that they can be used to generate code.

Table A.2: Imposition Times of constraints in this document

B Possible Implementations for the Configuration Steps

B.1 Alternative Approaches (Informative)

This chapter contains description of alternative approaches and information that is not part of the AUTOSAR, but can be helpful and give some guidance.

B.1.1 Alternative Configuration Editor Approaches

The EcuC Parameter Definition and ECU Configuration Values are designed to support a variety of different tooling approaches.

In the following, the different approaches that have been considered during the development of the specification are introduced. These tooling approaches are supported by EcuC Parameter Definition and ECU Configuration Values. Other approaches might be consistent with this specification, but have not been considered explicitly.

Tool suppliers have a high degree of freedom in the approach their tools may take to ECU Configuration.

ECU Configuration tools might consist of a single monolithic editor capable of manipulating all aspects of ECU Configuration, it could be a core tool framework that takes plug-in components to manipulate particular aspects of ECU Configuration, it might be a set of specialized tools each capable of configuring a particular type or subset of software modules or, probably more likely, software vendors could supply individual custom tools to configure only the code blocks that they supply (similar to microprocessor vendors providing specialized debuggers for their own micros).

Common to the different tool approaches is that each configuration editor shall be capable of reading an (possibly incomplete) ECU Configuration Values ARXML and writing back its modified configuration results in the same format.

The modification may include changed values of ECU Configuration values and added instances of containers with all included ECU Configuration Parameter Values (only for containers/parameters with variable multiplicity).

In every case, the ECU Configuration Values is expected to be the point of reference, the backbone of the process.

The sections below look at some possible tool forms and identify some of their strengths and weaknesses.

B.1.1.1 Custom Editors

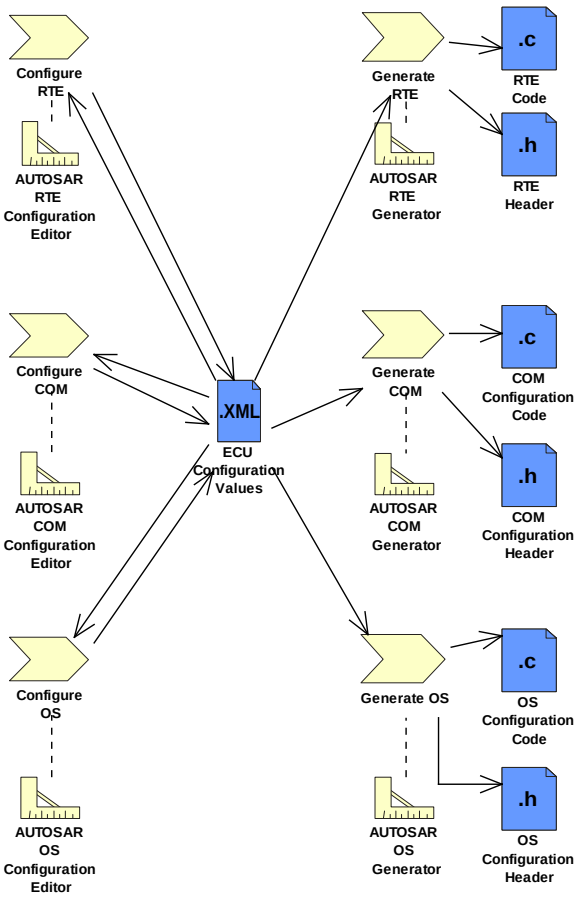

Figure B.1: Custom Editors and Generators

In the custom editors approach as shown in figure B.1, each BSW module is delivered bundled with a custom configuration editor and a generator (E.g. in figure B.1 the `AUTOSAR RTE Configuration Editor` and `AUTOSAR RTE Generator`).

These tools can be optimized to the particular task of configuring one BSW module and would likely be quite powerful. The complex dependencies between the BSW module configuration and other configuration items in the `ECU Configuration Values` could be expressed and supported in the tool.

Each vendor of a BSW module would need to provide a tool. System and ECU engineers would require a large number of tools to deal with the range of BSW modules. Each tool would probably have an individual look and feel and this could increase the training and experience required to become proficient.

B.1.1.2 Generic Tools

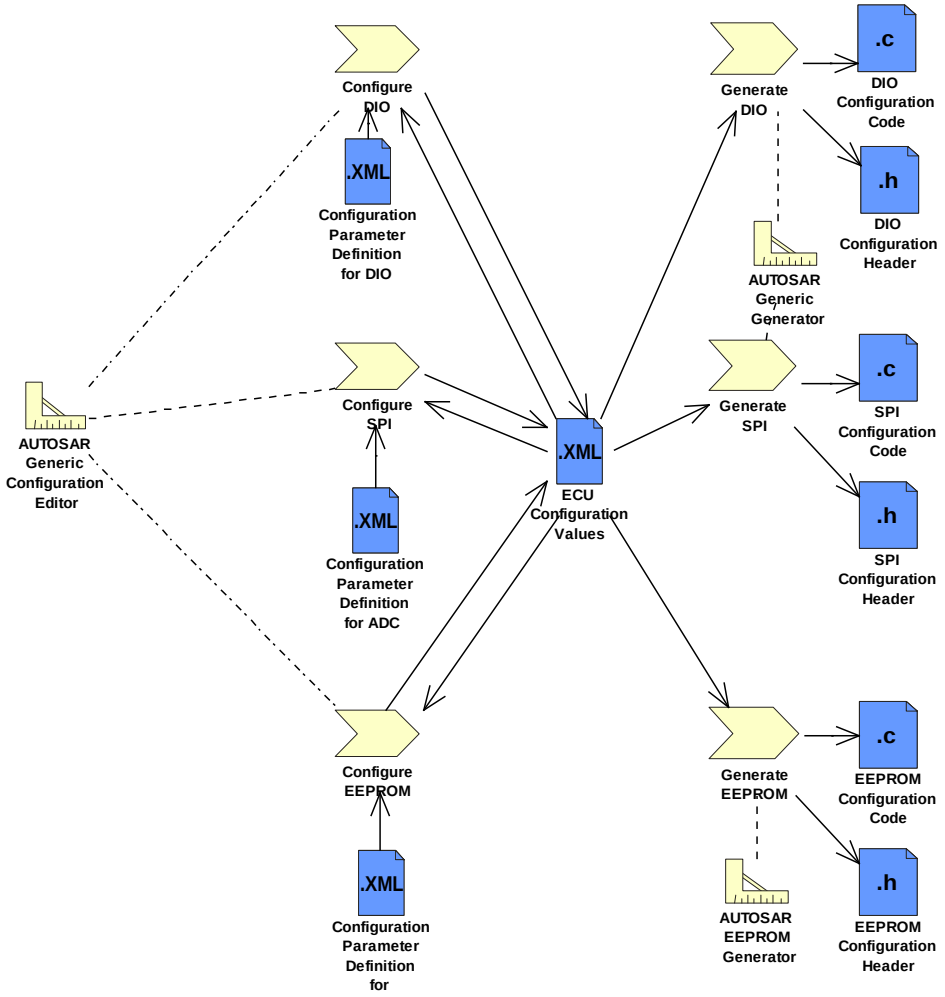

Figure B.2: Generic Configuration Editor

An AUTOSAR Generic Configuration Editor as shown in figure B.2 would be able to configure any parameter defined in Configuration Parameter Definitions. It would read those definitions and provide a generic interface to edit values for all parameters in the ECU Configuration Values.

It would only be able to resolve the relatively simple dependencies explicitly defined in the ECU Configuration Parameter Definition. Only a limited number of editors would be required, maybe only one, and the look and feel is less likely to vary greatly between generic tools.

Training and tooling costs are therefore likely to be lower. Examples of such tools that already exist are tresos, GENy, DAvE and MICROSAR. On the generation side, either a generic generator may be used, or custom generators for the individual modules.

B.1.1.3 Tools Framework

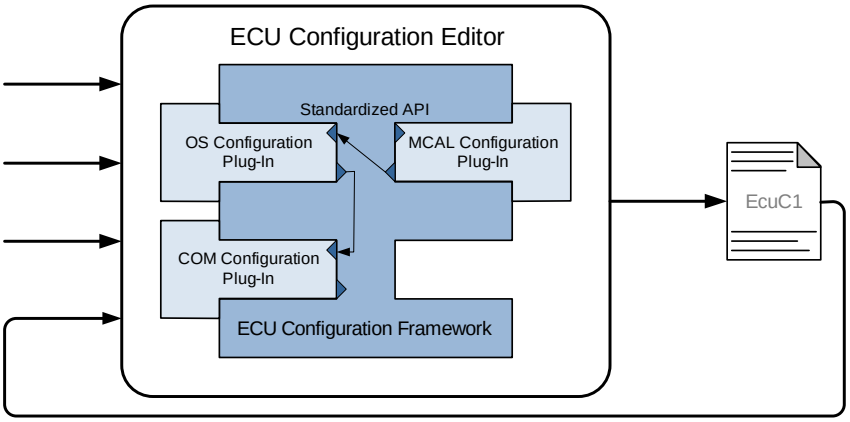

Figure B.3: Framework Tools

The tool framework as shown in figure B.3 is a cross between custom tools and generic tools where dedicated configuration steps (OS, COM, MCAL, etc.) are integrated as plug-ins into the common ECU Configuration framework.

The heart of the tool would be a framework that provides certain core services such as importing and exporting data from standard file formats, maintaining standard internal data structures and providing an HMI to the user. This ensures that the `ECU Configuration Values` is read, interpreted and written in a defined way.

The frame could also monitor and control the user / project work flow. It provides a low initial tooling and training investment. The power of the approach would be the ability to add plug-in modules that extend the core functionality.

These would allow very sophisticated features, potentially dedicated to one BSW module, to be added by individual suppliers. Additionally, it would be possible to add generic plug-ins that addressed a specific aspect of configuration across all BSW modules. This approach relies upon a standard framework: multiple framework standards could lead to high tool costs.

An example of this kind of tool is the LabVIEW test and measurement tool from National Instruments and the Eclipse framework.

B.1.2 Alternative Generation Approaches

As stated before, the `ECU Configuration Values` is the only configuration source that stores the configuration parameters for all modules of an AUTOSAR based ECU.

However, for several modules such as OS, existing configuration languages have already been established. Those languages probably will in future still be used for non-AUTOSAR systems. Thus, modules that are used both for AUTOSAR and non-AUTOSAR systems shall support different configuration languages in parallel.

This can be implemented in different ways, as shown in figure B.4.

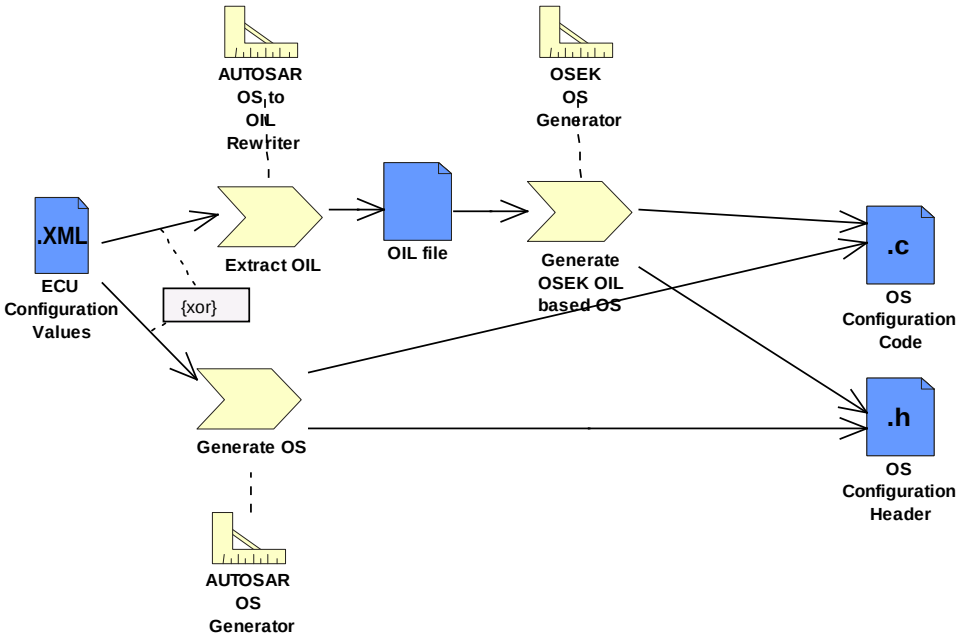

Figure B.4: Module generator implementation alternatives, example for OS

In a fully AUTOSAR based approach, the generator reads in the `ECU Configuration Values` and output the relevant source code directly in one step, supported by a `AUTOSAR OS Generator` in the example given.

To support reuse of existing generator tools, this single step can be split into two steps. Then the first activity is to extract all OS configuration information from the `ECU Configuration Values` using an `AUTOSAR OS to OIL Rewriter` and to store it in the format used by the legacy tools (`OIL file` in the example chosen).

The existing `OSEK OS Generator` may then read the intermediate format to generate the code. However, the intermediate format shall **not** be changed by any legacy configuration tools, since those changes would possibly make the overall configuration inconsistent, and since changes would be lost with the next generation of the intermediate format.

Thus, none of the activities (extract, generate) shall include any engineering step with decisions taken by an engineer. They are fully automatic, since all input driving these steps is included in the `ECU Configuration Values`.

C Glossary

The content of this appendix chapter is *informative* in nature and shall **not** be considered as *normative* content.

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 ([17]).

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: `PortPrototype`, `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.

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.

D Change History

The content of this appendix chapter is *informative* in nature and shall **not** be considered as *normative* content.

D.1 Change History for release R4.0.1

D.1.1 Renamed Meta-Model Elements for AUTOSAR Release 4.0

In the course of preparing AUTOSAR Release 4.0 some of the existing meta-model elements have been renamed for a better clarity and consistency with respect to other meta-mode elements.

D.1.2 Deleted SWS Items

SWS Item	*Rationale*
[ecuc_sws_3000]	Removed type specific value definitions.
[ecuc_sws_3001]	Removed type specific value definitions.
[ecuc_sws_3002]	Removed type specific value definitions.
[ecuc_sws_3003]	Removed type specific value definitions.
[ecuc_sws_3041]	Removed type specific value definitions.
[ecuc_sws_3005]	Removed type specific value definitions.
[ecuc_sws_1031]	The requirement from chapter B.1.1 has been changed to [TPS_ECUC_02124] because there were two requirements for this Id.
[ecuc_sws_2046]	Removed due to changes on the derived parameter structure.
[ecuc_sws_2048]	Removed due to changes on the derived parameter structure.
[ecuc_sws_2049]	Removed due to changes on the derived parameter structure.
[ecuc_sws_2050]	Removed due to changes on the derived parameter structure.
[ecuc_sws_2051]	Removed due to changes on the derived parameter structure.
[ecuc_sws_2052]	Removed due to changes on the derived parameter structure.
[ecuc_sws_2053]	Removed due to changes on the derived parameter structure.
[ecuc_sws_3022]	Removed due to changes on the derived parameter structure.
[ecuc_sws_3023]	Removed due to changes on the derived parameter structure.
[ecuc_sws_3024]	Removed due to changes on the derived parameter structure.
[ecuc_sws_3025]	Removed due to changes on the derived parameter structure.
[ecuc_sws_3026]	Removed due to changes on the derived parameter structure.
[ecuc_sws_1008]	Removed due to changes on the derived parameter structure.
[ecuc_sws_5004]	Removed due to changes on the derived parameter structure.
[ecuc_sws_5005]	Removed due to changes on the derived parameter structure.
[ecuc_sws_5006]	Removed due to changes on the derived parameter structure.
[ecuc_sws_2085]	Removed due to changes in Ecu Extract description.
[ecuc_sws_2086]	Removed due to changes in Ecu Extract description.
[ecuc_sws_2045]	EcuC Parameter Definitions can also be manually generated.
[ecuc_sws_2113]	currently not supported by the Variant Handling concept (aggregation of Primitives).
[ecuc_sws_2068]	Replaced requirement by [TPS_ECUC_02012].
[ecuc_sws_1000]	Replaced requirement by [TPS_ECUC_06038].

[ecuc_sws_1002]	Replaced with [TPS_ECUC_06007] and [TPS_ECUC_06008] for clarification of derivation rules.
[ecuc_sws_1003]	Replaced with [TPS_ECUC_06007] and [TPS_ECUC_06008] for clarification of derivation rules.
[ecuc_sws_1010]	Removed for clarification of derivation rules.
[ecuc_sws_1012]	Removed for clarification of derivation rules.

Table D.1: Deleted SWS Items

D.1.3 Changed SWS Items

SWS Item	*Heading*
[TPS_ECUC_02029]	Subclasses of EcucAbstractStringParamDef
[TPS_ECUC_02030]	Programming language identifier limitations
[TPS_ECUC_02031]	Restriction on the length of EcucLinkerSymbolDef values and defaultValue
[TPS_ECUC_02108]	Rule for the creation of #define symbols in the header file for parameters with the symbolicNameValue set to TRUE
[ecuc_sws_5001]	Refined because of unclear requirement.
[TPS_ECUC_03021]	EcucParameterDefs with EcucDerivationSpecification result in a EcucNumericalParamValue in the ECUC Value description
[TPS_ECUC_02047]	Derivation of parameter values
[TPS_ECUC_02087]	Creation of ServiceSwComponentTypes
[TPS_ECUC_02000]	Modeling of ECU Configuration Value and ECU Configuration Parameter Definition metamodels
[ecuc_sws_2084]	Incompatible inter module queries.
[ecuc_sws_6002]	Incompatible inter module queries.
[TPS_ECUC_01032]	Link time configuration
[TPS_ECUC_02030]	Programming language identifier limitations
[TPS_ECUC_02095]	VSMD refines the StMD
[TPS_ECUC_03007]	Attribute value stores the configuration value in XML-based description
[TPS_ECUC_03034]	Each parameter in an Ecuc Configuration Value description shall have a value
[TPS_ECUC_03010]	Parameters that are declared as optional in the ECU Configuration Definition may be left out in the ECU Configuration Value description
[TPS_ECUC_03040]	The value of an EcucNumericalParamValue shall be unambiguously an integer value
[TPS_ECUC_02107]	Values of parameters with the symbolicNameValue set to TRUE that are assigned by the configuration editor or module generator shall be stored in the XML file
[TPS_ECUC_02001]	Transformation of the ECU Configuration Value and ECU Configuration Parameter Definition metamodels to schema definitions

[TPS_ECUC_02002]	Generic structure of all AUTOSAR templates
[TPS_ECUC_06004]	AdminData field in ECU Configuration Parameter Definition XML file
[TPS_ECUC_02067]	Multiplicity of the to be chosen containers
[TPS_ECUC_02012]	Allowed choice of available to be chosen containers in the ECU Configuration Value description
[TPS_ECUC_02009]	Expression of optionality of containers, parameters and references
[TPS_ECUC_02029]	Subclasses of EcucAbstractStringParamDef
[TPS_ECUC_02087]	Creation of ServiceSwComponentTypes

Table D.2: Changed SWS Items

D.1.4 Added SWS Items

SWS Item	*Heading*
[TPS_ECUC_02110]	Variable lower and upper multiplicity in ECU Configuration Parameter definition
[TPS_ECUC_02111]	Variable default value in EcucBooleanParamDef
[TPS_ECUC_02112]	Variable default value in EcucAbstractStringParamDef
[ecuc_sws_2113]	Implement Variant Handling Concept.
[TPS_ECUC_02114]	Variable default value in EcucIntegerParamDef
[TPS_ECUC_02115]	Variable default value in EcucFloatParamDef
[TPS_ECUC_02116]	Variable min, max values in EcucIntegerParamDef
[TPS_ECUC_02117]	Variable min, max values in EcucFloatParamDef
[TPS_ECUC_02119]	Variable existence of container on value side
[TPS_ECUC_02120]	Variable subContainers
[TPS_ECUC_02121]	Variable parameterValues
[TPS_ECUC_02122]	Variable referenceValues
[TPS_ECUC_02118]	EcucAddInfoParamDef properties
[TPS_ECUC_02123]	The value of the parameter type EcucAddInfoParamDef
[TPS_ECUC_02124]	Tooling approaches that are supported by the ECUC parameter definition and ECUC Value description
[TPS_ECUC_02125]	Value of parameters with a defined derivation specification

[TPS_ECUC_02126]	Values for parameter types stored in the element EcucTextualParamValue
[TPS_ECUC_02127]	Possible values for EcucBooleanParamDef parameters
[TPS_ECUC_02128]	Formal description of the derivation
[TPS_ECUC_02129]	Informal description of the derivation
[TPS_ECUC_02130]	Standardized Module Definition package structure
[TPS_ECUC_06014]	Content of CompositionSwComponentType in the ECU Configuration
[TPS_ECUC_06015]	DESTINATION-REF in the VSMD
[TPS_ECUC_06016]	Countably infinite number of containers, parameters and references in the ECU Configuration Value description
[TPS_ECUC_06017]	Existence of upperMultiplicityInfinite and upperMultiplicity is mutually exclusive
[TPS_ECUC_06018]	Input and Output of the refvalue function
[TPS_ECUC_06019]	Output of the refvalue function if the EcucDefinitionElement points to a not existing element in the ECU Configuration Parameter Definition
[TPS_ECUC_06020]	Output of the refvalue function if no element in the ECU Configuration Value description is found
[TPS_ECUC_06021]	Input and Output of the deref function
[TPS_ECUC_06022]	Output of the deref function in case the first input parameter is a reference
[TPS_ECUC_06023]	Cases where the deref function reports an error
[TPS_ECUC_06024]	Input of the value function
[TPS_ECUC_06025]	Output of the value function
[TPS_ECUC_06026]	Cases where the value function reports an error
[TPS_ECUC_06027]	Input of the count function
[TPS_ECUC_06028]	Output of the count function
[TPS_ECUC_06029]	Output of the count function in case the input parameter set is empty
[TPS_ECUC_06030]	Invalid PduLength parameter value configuration
[TPS_ECUC_06031]	Interaction of Complex Driver with standardized AUTOSAR BSW modules
[TPS_ECUC_06032]	Min and max values in EcucIntegerParamDef
[TPS_ECUC_06033]	Min and max values in EcucFloatParamDef
[TPS_ECUC_06034]	Special float values
[TPS_ECUC_01032]	Link time configuration

[TPS_ECUC_06036]	Distinction of module definitions of Complex Drivers
[TPS_ECUC_06037]	apiServicePrefix attribute for Complex Driver modules
[TPS_ECUC_06038]	Rules to validate a BSW module implementation
[TPS_ECUC_06043]	EcucModuleDef categories
[TPS_ECUC_06044]	refinedModuleDef reference in the StMD
[TPS_ECUC_06035]	Regular expression
[TPS_ECUC_06006]	EcucLinkerSymbolDef properties
[TPS_ECUC_06007]	Elements defined in the StMD shall be present in the VSMD
[TPS_ECUC_06008]	lowerMultiplicity and upperMultiplicity of elements in the VSMD
[TPS_ECUC_06009]	Existence of a parameter in the Ecuc Parameter Value description in case the upperMultiplicity of the parameter definition is zero
[TPS_ECUC_06010]	Existence of a parameter in the Ecuc Parameter Value description in case the lowerMultiplicity of the parameter definition is bigger than zero
[TPS_ECUC_06011]	Missing parameters in the Ecuc Parameter Value description
[TPS_ECUC_06012]	Parameters without parameter definitions in the Ecuc Parameter Value description
[TPS_ECUC_06013]	Number of parameters in the Ecuc Parameter Value description

Table D.3: Added SWS Items

D.2 Change History for release R4.0.2

D.2.1 Changed SWS Items

SWS Item	*Heading*
[TPS_ECUC_02072]	Signed EcucIntegerParamDef value range
[TPS_ECUC_02095]	VSMD refines the StMD

Table D.4: Changed SWS Items

D.2.2 Added SWS Items

SWS Item	*Heading*
[TPS_ECUC_02131]	Origin information in literal definitions

[TPS_ECUC_06039]	BswImplementation and vendorSpecificModuleDef shall be known by tools
[TPS_ECUC_06040]	EcucValueCollection is the input for tools
[TPS_ECUC_06041]	Tools shall respect the EcucModuleConfigurationValues elements that are referenced by the EcucValueCollection
[TPS_ECUC_06042]	Tools interaction with EcucModuleConfigurationValues
[TPS_ECUC_06043]	EcucModuleDef categories
[TPS_ECUC_06044]	refinedModuleDef reference in the StMD
[TPS_ECUC_06045]	min, max values of parameters in the VSMD in case that the min or max value in the StMD is set to infinite

Table D.5: Added SWS Items

D.3 Change History for release R4.0.2

D.3.1 Deleted SWS Items

SWS Item	*Rationale*
[ecuc_sws_5001]	Removed because it was unclear.

Table D.6: Deleted SWS Items

D.3.2 Changed SWS Items

SWS Item	*Heading*
[TPS_ECUC_02041]	EcucForeignReferenceDef properties
[TPS_ECUC_02082]	Specification of the destinationType in a EcucInstanceReferenceDef
[TPS_ECUC_02083]	Specification of the destinationContext in a EcucInstanceReferenceDef
[TPS_ECUC_06002]	Removal of standardized EcucEnumerationLiteralDefs in the VSMD
[TPS_ECUC_06015]	DESTINATION-REF in the VSMD
[TPS_ECUC_06025]	Output of the value function
[TPS_ECUC_06037]	apiServicePrefix attribute for Complex Driver modules
[TPS_ECUC_02108]	Rule for the creation of #define symbols in the header file for parameters with the symbolicNameValue set to TRUE

Table D.7: Changed SWS Items

D.3.3 Added SWS Items

SWS Item	*Heading*
[TPS_ECUC_02132]	The postBuildChangeable attribute shall only be set to true for containers located within a multipleConfigurationContainer
[TPS_ECUC_02133]	upperMultiplicity of a multipleConfigurationContainer
[TPS_ECUC_06046]	Vendor specific reference definition with no counterpart in the STMD
[TPS_ECUC_06047]	References in the ECUC Parameter Value description with reference definitions that refer to container definitions in the same module definition
[TPS_ECUC_06048]	References in the ECUC Parameter Value description with reference definitions that refer to container definitions in different module definitions
[TPS_ECUC_06049]	Restriction of supportedConfigVariants in the VSMD
[TPS_ECUC_06050]	supportedConfigVariants in the VSMD in case VariantPostBuild is supported in the StMD
[TPS_ECUC_06051]	ImplementationConfigClass of an EcucParameterDef or EcucAbstractReferenceDef in VSMD
[TPS_ECUC_06052]	Supported configuration variants in the VSMD
[TPS_ECUC_06053]	VSMD Configuration variant “VariantPreCompile”
[TPS_ECUC_06054]	VSMD Configuration variant “VariantLinkTime”
[TPS_ECUC_06055]	VSMD Configuration variant “VariantPostBuildLoadable”
[TPS_ECUC_06056]	VSMD Configuration variant “VariantPostBuildSelectable”
[TPS_ECUC_06057]	Input of the strValue function
[TPS_ECUC_06058]	Output of the strValue function
[TPS_ECUC_06059]	Cases where the strValue function reports an error
[TPS_ECUC_06060]	Input of the valueAt function
[TPS_ECUC_06061]	Output of the valueAt function
[TPS_ECUC_06062]	Cases where the valueAt function reports an error
[TPS_ECUC_06063]	Input of the strValueAt function
[TPS_ECUC_06064]	Output of the strValueAt function
[TPS_ECUC_06065]	Cases where the strValueAt function reports an error
[TPS_ECUC_06066]	Order of Container-, Parameter- and Reference-Values
[TPS_ECUC_06067]	Sorting criteria for Containers on the Values side
[TPS_ECUC_06068]	Sorting criteria for References on the Values side

[TPS_ECUC_06069]	Sorting criteria for Parameters on the Values side
[TPS_ECUC_06070]	Sorting of Ecu Configuration Parameter Definitions
[TPS_ECUC_06071]	ECU Configuration Editor shall be able to read parameter values in any order
[TPS_ECUC_06072]	Container-, Parameter-, and Reference-Values with requiresIndex set to true
[TPS_ECUC_06073]	The ECU Configuration Editor shall be able to work with arbitrary package structures
[TPS_ECUC_06074]	Invalid configuration due to symbolic name values

Table D.8: Added SWS Items

D.3.4 Added Constraints

Number	Heading
[constr_3022]	EcucModuleDef category restriction
[constr_3023]	Usage of apiServicePrefix

Table D.9: Added Constraints in R4.0.3

D.4 Change History for release R4.1.1

D.4.1 Deleted SWS Items

SWS Item	*Rationale*

Table D.10: Deleted SWS Items

D.4.2 Changed SWS Items

SWS Item	*Heading*
[TPS_ECUC_06067]	Sorting criteria for Containers on the Values side
[TPS_ECUC_06072]	Container-, Parameter-, and Reference-Values with requiresIndex set to true
[TPS_ECUC_02039]	References between containers are established with the EcucReferenceDef
[TPS_ECUC_02040]	EcucChoiceReferenceDef properties
[TPS_ECUC_06051]	ImplementationConfigClass of an EcucParameterDef or EcucAbstractReferenceDef in VSMD

Table D.11: Changed SWS Items

D.4.3 Added SWS Items

SWS Item	*Heading*
[TPS_ECUC_02134]	requiresIndex setting in the VSMD
[TPS_ECUC_02135]	Validation of EcucValidationCondition
[TPS_ECUC_02136]	Validation of multiple EcucValidationConditions
[TPS_ECUC_02137]	EcucValidationConditions from the StMD shall be taken over to the VSMD.
[TPS_ECUC_02138]	Addition of vendor specific EcucValidationConditions
[TPS_ECUC_02139]	Definition of configuration classes for all CDD configuration parameters and references
[TPS_ECUC_02141]	Variable reference `EcucValueCollection.ecucValue`
[TPS_ECUC_02142]	Variable value of `EcucNumericalParamValue.value`
[TPS_ECUC_08001]	Configuration class of parameters and references within postBuildChangeable containers
[TPS_ECUC_08002]	Introduction of new EcucParamConfContainerDef instances in a post-build loadable configuration set
[TPS_ECUC_08003]	Usage of postBuildChangeable attribute is independent of aggregated subContainers
[TPS_ECUC_08004]	Changing of values and multiplicities of EcucParameterValues at post-build time
[TPS_ECUC_08005]	postBuildChangeable attribute in the VSMD in case it is not defined in the StMD
[TPS_ECUC_08006]	postBuildChangeable attribute in the VSMD in case it is set to false in the StMD
[TPS_ECUC_08007]	postBuildChangeable attribute in the VSMD in case it is set to true in the StMD
[TPS_ECUC_08008]	Usage of the multiple configuration container in EcucModuleDefs with supportedConfigVariant of VariantPostBuild
[TPS_ECUC_08009]	Names of containers inside a multiple configuration set
[TPS_ECUC_08010]	Ticks in the Ecuc Parameter Value description
[TPS_ECUC_08011]	Pattern for creating a C symbol used by the EcuM/BswM to initialize post-build Bsw modules
[TPS_ECUC_02143]	Optional configuration of Production Error and Extended Production Error reporting
[TPS_ECUC_02144]	Definition of supported config variants for CDD

Table D.12: Added SWS Items

D.4.4 Added Constraints

[constr_3509]	Applicability of scope attribute
[constr_5500]	Applicability of postBuildChangeable attribute
[constr_5501]	EcucParameterValues and EcucAbstractReferenceValues in EcucContainerValues that exist in multiple configuration sets
[constr_5502]	EcucParameterValues of type EcucFunctionNameDef
[constr_5503]	symbolicNameValue parameters in post-build configuration sets
[constr_5504]	Removing an instance of the EcucContainerDef in post-build time
[constr_5505]	Configuration class of the elements of the EcucQueryExpression

Table D.13: Added Constraints in R4.1.1

D.5 Change History for release R4.1.2

D.5.1 Deleted SWS Items

SWS Item	*Rationale*
[TPS_ECUC_06002]	Removal of standardized EcucEnumerationLiteralDefs in the VSMD
[TPS_ECUC_01036]	Migrated to TR_METH_01114
[TPS_ECUC_01027]	Migrated to TR_METH_01115
[TPS_ECUC_01028]	Migrated to TR_METH_01116
[TPS_ECUC_01031]	Equivalent to TR_METH_01095
[TPS_ECUC_01032]	Equivalent to TR_METH_01098
[TPS_ECUC_04006]	Equivalent to TR_METH_01104
[TPS_ECUC_04007]	Equivalent to TR_METH_01107
[TPS_ECUC_01029]	Migrated to TR_METH_01117
[TPS_ECUC_01030]	Equivalent to TR_METH_01089
[TPS_ECUC_04000]	Deprecated
[TPS_ECUC_04001]	Relies on deprecated chapter
[TPS_ECUC_02132]	Contradictory to the variant handling approach described in chapter 2.4.7

Table D.14: Deleted SWS Items

D.5.2 Changed SWS Items

SWS Item	*Heading*
[TPS_ECUC_04002]	ECU Configuration Editor shall be able to merge `ECU Configuration Value descriptions`

Table D.15: Changed SWS Items

D.5.3 Added SWS Items

SWS Item	*Heading*
[TPS_ECUC_06075]	`EcucFunctionNameDef` shall represent a valid C Identifier

Table D.16: Added SWS Items

D.6 Change History for release R4.1.3

D.6.1 Deleted SWS Items

SWS Item	*Heading*
[TPS_ECUC_02022]	Configuration variants of a BSW module in the ECU Configuration Parameter Definition

Table D.17: Deleted SWS Items

D.6.2 Changed SWS Items

SWS Item	*Heading*
[TPS_ECUC_02095]	VSMD refines the StMD
[TPS_ECUC_02140]	Mandatory configuration of CddConfigSet for post build configured CDD

Table D.18: Changed SWS Items

D.6.3 Added SWS Items

SWS Item	*Heading*
[TPS_ECUC_06076]	Use cases where the reference `refinedModuleDef` is mandatory
[TPS_ECUC_06077]	Use cases where the reference `refinedModuleDef` is optional

Table D.19: Changed SWS Items

D.6.4 Added Constraints

[constr_3091]	Multiplicity of implementationConfigClass
[constr_3092]	Usage of EcucImplementationConfigurationClass.configVariant and EcucImplementationConfigurationClass.configClass attributes

Table D.20: Added Constraints in R4.1.3

D.7 Change History for release R4.2.1

D.7.1 Added Specification Items in 4.2.1

Id	Heading
[TPS_ECUC_06078]	`EcucUriReferenceDef` properties
[TPS_ECUC_06079]	`EcucDestinationUriPolicy.destinationUriNestingContract` is set to `targetContainer`
[TPS_ECUC_06080]	`EcucDestinationUriPolicy.destinationUriNestingContract` is set to `leafOfTargetContainer`
[TPS_ECUC_06081]	`EcucDestinationUriPolicy.destinationUriNestingContract` is set to `vertexOfTargetContainer`
[TPS_ECUC_08012]	Module support for post-build variants
[TPS_ECUC_08013]	Different number of `EcucContainerDef` instances in different post-build variants
[TPS_ECUC_08014]	Usage of `postBuildVariantMultiplicity` attribute is independent of aggregated `subContainer`s
[TPS_ECUC_08015]	Different number of `EcucCommonAttributes`s instances in different post-build variants
[TPS_ECUC_08016]	Different values of `EcucCommonAttributes`s instances in different post-build variants
[TPS_ECUC_08017]	Derivation of information from parameter values bound at `PreCompile` time
[TPS_ECUC_08018]	Derivation of information from parameter values bound at `Link` time
[TPS_ECUC_08019]	Derivation of information from parameter values bound at `PostBuild` time
[TPS_ECUC_08021]	The value of the `EcucModuleDef.postBuildVariantSupport` attribute in the VSMD in case it is not defined in the StMD
[TPS_ECUC_08025]	The value of the `EcucContainerDef.postBuildVariantMultiplicity` attribute in the VSMD in case it is not defined in the StMD
[TPS_ECUC_08026]	The value of the `EcucContainerDef.postBuildVariantMultiplicity` attribute in the VSMD in case it is set to `false` in the StMD
[TPS_ECUC_08027]	The value of the `EcucContainerDef.postBuildVariantMultiplicity` in the VSMD in case it is set to `true` in the StMD
[TPS_ECUC_08028]	The value of the `EcucParameterDef.postBuildVariantMultiplicity` and the `EcucAbstractReferenceDef.postBuildVariantMultiplicity` attributes in the VSMD in case they are not defined in the StMD

[TPS_ECUC_08029]	The value of the EcucParameterDef.postBuildVariantValue and the EcucAbstractReferenceDef.postBuildVariantValue attributes in the VSMD in case they are not defined in the StMD
[TPS_ECUC_08030]	The value of the EcucParameterDef.postBuildVariantValue and the EcucAbstractReferenceDef.postBuildVariantValue attributes in the VSMD in case they are set to false in the StMD
[TPS_ECUC_08031]	The value of the EcucParameterDef.postBuildVariantMultiplicity and the EcucAbstractReferenceDef.postBuildVariant-Multiplicity attributes in the VSMD in case they are set to false in the StMD
[TPS_ECUC_08032]	The value of the EcucParameterDef.postBuildVariantValue and the EcucAbstractReferenceDef.postBuildVariantValue attributes in the VSMD in case they are set to true in the StMD
[TPS_ECUC_08033]	The value of the EcucParameterDef.postBuildVariantMultiplicity and the EcucAbstractReferenceDef.postBuildVariant-Multiplicity attributes in the VSMD in case they are set to true in the StMD
[TPS_ECUC_08034]	Different values of EcucCommonAttributess instances in different configuration times
[TPS_ECUC_08035]	Different number of instances of EcucCommonAttributess in different configuration times
[TPS_ECUC_08036]	The value of the EcucParameterDef.valueConfigClass and the Ecuc-AbstractReferenceDef.valueConfigClass attributes in the VSMD in case they are not defined in the StMD
[TPS_ECUC_08037]	The value of the EcucParameterDef.multiplicityConfigClass and the EcucAbstractReferenceDef.multiplicityConfigClass attributes in the VSMD in case they are not defined in the StMD
[TPS_ECUC_08038]	The value of the EcucParameterDef.valueConfigClass and the Ecuc-AbstractReferenceDef.valueConfigClass attributes in the VSMD in case they are defined in the StMD
[TPS_ECUC_08039]	The value of the EcucParameterDef.multiplicityConfigClass and the EcucAbstractReferenceDef.multiplicityConfigClass attributes in the VSMD in case they are defined in the StMD
[TPS_ECUC_08041]	The value of the EcucModuleDef.postBuildVariantSupport attribute in the VSMD in case it set to false in the StMD
[TPS_ECUC_08042]	The value of the EcucModuleDef.postBuildVariantSupport attribute in the VSMD in case it is set to true in the StMD
[TPS_ECUC_08043]	The number of EcucContainerValue instances in post-build time updated ECU configurations
[TPS_ECUC_08044]	The number of EcucContainerValue instances in different post-build variants
[TPS_ECUC_08045]	The value of EcucParameterValue instances in post-build time updated ECU configurations
[TPS_ECUC_08046]	The value of EcucParameterValue instances in different post-build variants
[TPS_ECUC_08047]	The number of EcucParameterValue instances in post-build time updated ECU configurations
[TPS_ECUC_08048]	The number of EcucParameterValue instances in different post-build variants
[TPS_ECUC_08049]	The value of EcucAbstractReferenceValue instances in post-build time updated ECU configurations
[TPS_ECUC_08050]	The value of EcucAbstractReferenceValue instances in different post-build variants
[TPS_ECUC_08051]	The number of EcucAbstractReferenceValue instances in post-build time updated ECU configurations

[TPS_ECUC_08052]	The number of EcucAbstractReferenceValue instances in different post-build variants

Table D.21: Added Traceables in 4.2.1

D.7.2 Changed Specification Items in 4.2.1

Id	Heading
[TPS_ECUC_01021]	Values derived from the ECU extract of the system configuration.
[TPS_ECUC_02016]	Configuration class of parameter and reference definitions
[TPS_ECUC_02019]	Configuration class “PostBuild”
[TPS_ECUC_02056]	Derivation of information from Link parameters
[TPS_ECUC_02057]	Derivation of information from PostBuild parameters
[TPS_ECUC_02058]	Derivation of information from PreCompile parameters
[TPS_ECUC_02097]	Supported configuration variants in the StMD and the VSMD
[TPS_ECUC_02098]	StMD Configuration variant "VariantPreCompile"
[TPS_ECUC_02099]	StMD Configuration variant "VariantLinkTime"
[TPS_ECUC_02100]	StMD Configuration variant "VariantPostBuild"
[TPS_ECUC_02101]	EcucAbstractConfigurationClass usage
[TPS_ECUC_02102]	Configuration class selection for parameters and references for supported configuration variants
[TPS_ECUC_02139]	Definition of configuration classes for all CDD configuration parameters and references
[TPS_ECUC_06001]	shortName of a VSMD module
[TPS_ECUC_06008]	lowerMultiplicity and upperMultiplicity of elements in the VSMD
[TPS_ECUC_06046]	Vendor specific reference definition with no counterpart in the STMD
[TPS_ECUC_08000]	Different number of EcucContainerDef instances in different configuration times
[TPS_ECUC_08002]	Introduction of new EcucParamConfContainerDef instances in updated post-build configuration
[TPS_ECUC_08003]	Usage of multiplicityConfigClass.configClass attribute is independent of its aggregated subContainers
[TPS_ECUC_08005]	The value of the EcucContainerDef.multiplicityConfigClass attribute in the VSMD in case it is not defined in the StMD
[TPS_ECUC_08006]	The value of the EcucContainerDef.multiplicityConfigClass attribute in the VSMD in case it is defined in the StMD
[TPS_ECUC_08011]	Pattern for creating a C symbol used by the EcuM/BswM to initialize BSW modules with different post-build variants

Table D.22: Changed Traceables in 4.2.1

D.7.3 Deleted Specification Items in 4.2.1

Id	Heading
[TPS_ECUC_02055]	Derivation of information from AUTOSAR Templates
[TPS_ECUC_02076]	“NOAffect” affection
[TPS_ECUC_02077]	“PCAffectsLT” affection
[TPS_ECUC_02078]	“PCAffectsPB” affection
[TPS_ECUC_02079]	“PCAffectsLTAndPB” affection
[TPS_ECUC_02080]	“LTAffectsPB” affection

[TPS_ECUC_02081]	Parameters or references which are affected may be referenced with the `affected` reference
[TPS_ECUC_02091]	`multipleConfigurationContainer` approach
[TPS_ECUC_02092]	`multipleConfigurationContainer` allows several `EcucContainerValue` elements in the ECU Configuration
[TPS_ECUC_02104]	Valid configuration set names
[TPS_ECUC_02105]	Uniqueness of configuration set names
[TPS_ECUC_02133]	`upperMultiplicity` of a `multipleConfigurationContainer`
[TPS_ECUC_02140]	Mandatory configuration of `CddConfigSet` for post build configured CDD
[TPS_ECUC_03042]	Definition of multiple configuration sets
[TPS_ECUC_03043]	Occurrence of multiple configuration containers in the ECUC Value description
[TPS_ECUC_03044]	Name of the configuration set
[TPS_ECUC_03045]	Parameter value description structure underneath the multiple configuration container
[TPS_ECUC_03046]	Values of pre-compile time and link time parameters in different configuration sets
[TPS_ECUC_03047]	`EcucReferenceValue` in multiple configuration sets
[TPS_ECUC_06050]	`supportedConfigVariant`s in the VSMD in case `VariantPostBuild` is supported in the StMD
[TPS_ECUC_06051]	ImplementationConfigClass of an `EcucParameterDef` or `EcucAbstractReferenceDef` in VSMD
[TPS_ECUC_06052]	Supported configuration variants in the VSMD
[TPS_ECUC_06053]	VSMD Configuration variant “VariantPreCompile”
[TPS_ECUC_06054]	VSMD Configuration variant “VariantLinkTime”
[TPS_ECUC_06055]	VSMD Configuration variant “VariantPostBuildLoadable”
[TPS_ECUC_06056]	VSMD Configuration variant “VariantPostBuildSelectable”
[TPS_ECUC_08001]	Configuration class of parameters and references within `postBuildChangeable` containers.
[TPS_ECUC_08004]	Changing of values and multiplicities of `EcucParameterValue`s at post-build time
[TPS_ECUC_08007]	`postBuildChangeable` attribute in the VSMD in case it is set to true in the StMD
[TPS_ECUC_08008]	Usage of the multiple configuration container in `EcucModuleDef`s with `supportedConfigVariant` of `VariantPostBuild`
[TPS_ECUC_08009]	Names of containers inside a multiple configuration set

Table D.23: Deleted Traceables in 4.2.1

D.7.4 Added Constraints in 4.2.1

Id	Heading
[constr_3119]	Necessary content of `EcucDestinationUriDef`s that are referenced by an `EcucContainerDef`
[constr_3120]	Applicable attributes when `EcucDestinationUriPolicy.destinationUriNestingContract` is set to `targetContainer`
[constr_5015]	Multiplicity of `multiplicityConfigClass`
[constr_5506]	Applicability of `postBuildVariantMultiplicity` attribute
[constr_5507]	Value of `EcucContainerDef.postBuildVariantMultiplicity` if `postBuildVariantSupport` is set to `false`
[constr_5508]	Applicability of `postBuildVariantMultiplicity` attribute

[constr_5509]	Value of postBuildVariantMultiplicity if postBuildVariantSupport is set to false
[constr_5510]	Value of postBuildVariantValue if postBuildVariantSupport is set to false
[constr_5512]	postBuildVariantValue attribute of symbolicNameValue parameters
[constr_5514]	Applicability of the multiplicityConfigClass attribute
[constr_5520]	valueConfigClass attribute of symbolicNameValue parameters
[constr_5521]	multiplicityConfigClass attribute of symbolicNameValue parameters
[constr_5522]	postBuildVariantMultiplicity attribute of symbolicNameValue parameters
[constr_5523]	Allowed configClasses for paired configVariants

Table D.24: Added Constraints in 4.2.1

D.7.5 Changed Constraints in 4.2.1

Id	Heading
[constr_3091]	Multiplicity of valueConfigClass
[constr_3092]	Usage of configVariant and configClass attributes
[constr_5500]	Applicability of the multiplicityConfigClass attribute
[constr_5502]	Introduction of new EcucParameterValues of type EcucFunctionNameDef at post-build time
[constr_5504]	Removing an instance of the EcucContainerDef at post-build time

Table D.25: Changed Constraints in 4.2.1

D.7.6 Deleted Constraints in 4.2.1

Id	Heading
[constr_5501]	EcucParameterValues and EcucAbstractReferenceValues in EcucContainerValues that exist in multiple configuration sets
[constr_5503]	symbolicNameValue parameters in post-build configuration sets

Table D.26: Deleted Constraints in 4.2.1

D.8 Change History for release R4.2.2

D.8.1 Added Specification Items in 4.2.2

Id	Heading
[TPS_ECUC_08053]	AUTOSAR release version in VSMD
[TPS_ECUC_08054]	Semantic of an optional parameter that is not present in the ECU Configuration Value description

Table D.27: Added Traceables in 4.2.2

D.8.2 Changed Specification Items in 4.2.2

Id	Heading
[TPS_ECUC_01001]	lowerMultiplicity and upperMultiplicity of modules in the VSMD
[TPS_ECUC_02072]	Signed EcucIntegerParamDef value range
[TPS_ECUC_02108]	Rule for the creation of #define symbols in the header file for parameters with the symbolicNameValue set to TRUE
[TPS_ECUC_03027]	EcucReferenceValue provides the mechanism to reference model elements that are Referrable
[TPS_ECUC_03033]	EcucInstanceReferenceValue provides the mechanism to reference an instance of a prototype
[TPS_ECUC_03034]	Each parameter in an ECU Configuration Value description shall have a value
[TPS_ECUC_06008]	lowerMultiplicity and upperMultiplicity of elements in the VSMD

Table D.28: Changed Traceables in 4.2.2

D.8.3 Deleted Specification Items in 4.2.2

none

D.8.4 Added Constraints in 4.2.2

Id	Heading
[constr_3200]	Restriction on values of EcucDefinitionElement.relatedTraceItem in the VSMD
[constr_3217]	Symbolic name reference shall point only to containers with a symbolic name value defined

Table D.29: Added Constraints in 4.2.2

D.8.5 Changed Constraints in 4.2.2

Id	Heading
[constr_3022]	EcucModuleDef category restriction
[constr_3023]	Usage of apiServicePrefix
[constr_5505]	Configuration class of the elements of the EcucQueryExpression
[constr_5506]	Applicability of postBuildVariantMultiplicity attribute
[constr_5507]	Value of EcucContainerDef.postBuildVariantMultiplicity if postBuildVariantSupport is set to false
[constr_5509]	Value of postBuildVariantMultiplicity if postBuildVariantSupport is set to false
[constr_5510]	Value of postBuildVariantValue if postBuildVariantSupport is set to false

Table D.30: Changed Constraints in 4.2.2

D.8.6 Deleted Constraints in 4.2.2

none

D.9 Change History for release R4.3.0

D.9.1 Added Specification Items in 4.3.0

Id	Heading
[TPS_ECUC_02145]	Attribute requiresSymbolicNameValue
[TPS_ECUC_02146]	Symbolic Name Reference properties
[TPS_ECUC_02147]	Introducing new post build variants at post build configuration time
[TPS_ECUC_06082]	Definition of interval type for EcucFloatParamDef.min and EcucFloat-ParamDef.max
[TPS_ECUC_06083]	Attribute EcucFloatParamDef.min.intervalType is not defined
[TPS_ECUC_06084]	Attribute EcucFloatParamDef.max.intervalType is not defined
[TPS_ECUC_06085]	Ordering of MetaDataItems of an MetaDataType
[TPS_ECUC_06086]	Relevance of the order of MetaDataItems of an MetaDataType

Table D.31: Added Traceables in 4.3.0

D.9.2 Changed Specification Items in 4.3.0

Id	Heading
[TPS_ECUC_08042]	The value of the EcucModuleDef.postBuildVariantSupport attribute in the VSMD in case it is set to true in the StMD

Table D.32: Changed Traceables in 4.3.0

D.9.3 Deleted Specification Items in 4.3.0

none

D.9.4 Added Constraints in 4.3.0

Id	Heading
[constr_3228]	EcucSymbolicNameReferenceDef presupposes requiresSymbolicName-Value set to true
[constr_3233]	EcucModuleDef that relies on EcucCommonAttributes with valueConfig-Class set to Link/PostBuild of another EcucModuleDef
[constr_3234]	EcucModuleDef that relies on EcucCommonAttributes with multiplicity-ConfigClass set to Link/PostBuild of another EcucModuleDef
[constr_3235]	EcucModuleDef that relies on EcucContainerDefs with multiplicity-ConfigClass set to Link/PostBuild of another EcucModuleDef
[constr_3236]	EcucModuleDef that relies on EcucCommonAttributes with postBuild-VariantValue set to true of another EcucModuleDef

[constr_3237]	EcucModuleDef that relies on EcucCommonAttributes with postBuildVariantMultiplicity set to true of another EcucModuleDef
[constr_3238]	EcucModuleDef that relies on EcucContainerDef with postBuildVariantMultiplicity set to true of another EcucModuleDef
[constr_3307]	ShortNames of PredefinedVariants referenced by EcucPostBuildVariantRefs

Table D.33: Added Constraints in 4.3.0

D.9.5 Changed Constraints in 4.3.0

Id	Heading
[constr_3217]	Symbolic name reference shall point only to containers with a symbolic name value defined

Table D.34: Changed Constraints in 4.3.0

D.9.6 Deleted Constraints in 4.3.0

none

D.10 Change History for release R4.3.1

D.10.1 Added Specification Items in 4.3.1

Number	Heading
[TPS_ECUC_06087]	INF and -INF allowed as defaultValue in EcucFloatParamDef

Table D.35: Added Specification Items in 4.3.1

D.10.2 Changed Specification Items in 4.3.1

Number	Heading
[TPS_ECUC_06074]	Invalid configuration due to symbolic name values

Table D.36: Changed Specification Items in 4.3.1

D.10.3 Deleted Specification Items in 4.3.1

none

D.10.4 Added Constraints in 4.3.1

none

D.10.5 Changed Constraints in 4.3.1

none

D.10.6 Deleted Constraints in 4.3.1

none

D.11 Change History for release R4.4.0

D.11.1 Added Specification Items in 4.4.0

Number	Heading
[TPS_ECUC_02148]	Replacement for `EcucSymbolicNameReferenceDef`

Table D.37: Added Specification Items in 4.4.0

D.11.2 Changed Specification Items in 4.4.0

Number	Heading
[TPS_ECUC_02063]	Parameters with `symbolicNameValue` = true
[TPS_ECUC_02147]	Introducing new post build variants at post build configuration time
[TPS_ECUC_03028]	`EcucReferenceValue` describes `EcucReferenceDef`s, `EcucChoiceReferenceDef`s, and `EcucForeignReferenceDef`s in the Ecu Configuration Value description
[TPS_ECUC_03037]	The `shortName` of the referenced container provides the symbolic name in the implementation
[TPS_ECUC_06012]	Parameters without parameter definitions in the Ecuc Parameter Value description

Table D.38: Changed Specification Items in 4.4.0

D.11.3 Deleted Specification Items in 4.4.0

Number	Heading
[TPS_ECUC_02032]	EcucSymbolicNameReferenceDef properties
[TPS_ECUC_02145]	Attribute requiresSymbolicNameValue
[TPS_ECUC_03036]	EcucSymbolicNameReferenceDef translates to a EcucReference-Value in the ECU Configuration Value description

Table D.39: Deleted Specification Items in 4.4.0

D.11.4 Added Constraints in 4.4.0

Number	Heading
[constr_3449]	Impact of postBuildVariantUsed value set to FALSE
[constr_3450]	postBuildVariantUsed value in case of post build VariationPoints
[constr_3451]	EcucModuleConfigurationValues.postBuildVariantUsed value setting restriction in case postBuildVariantSupport is set to TRUE
[constr_3452]	EcucModuleConfigurationValues.postBuildVariantUsed value setting restriction in case postBuildVariantSupport is set to FALSE

Table D.40: Added Constraints in 4.4.0

D.11.5 Changed Constraints in 4.4.0

none

D.11.6 Deleted Constraints in 4.4.0

Number	Heading
[constr_3228]	EcucSymbolicNameReferenceDef presupposes requiresSymbolicName-Value set to true

Table D.41: Deleted Constraints in 4.4.0

D.12 Change History for release R19-11

D.12.1 Added Specification Items in 19-11

Number	Heading
[TPS_ECUC_06088]	Specification of the destinationType format in a EcucForeign-ReferenceDef

Table D.42: Added Specification Items in 19-11

D.12.2 Changed Specification Items in 19-11

Number	Heading
[TPS_ECUC_01001]	lowerMultiplicity and upperMultiplicity of modules in the VSMD
[TPS_ECUC_01005]	Origin attribute of parameters in the VSMD that are taken over from the StMD
[TPS_ECUC_01007]	min, max values of parameters in the VSMD
[TPS_ECUC_01025]	Generate and extract activities are fully automatic
[TPS_ECUC_01035]	UUID of elements in the VSMD that are taken over from the StMD
[TPS_ECUC_04004]	Iterative development of the ECU Configuration Value description
[TPS_ECUC_06010]	Existence of a parameter in the Ecuc Parameter Value description in case the lowerMultiplicity of the parameter definition is bigger than zero
[TPS_ECUC_06031]	Interaction of Complex Driver with standardized AUTOSAR BSW modules
[TPS_ECUC_06032]	Min and max values in EcucIntegerParamDef
[TPS_ECUC_06033]	Min and max values in EcucFloatParamDef

Table D.43: Changed Specification Items in 19-11

D.12.3 Deleted Specification Items in 19-11

Number	Heading
[TPS_ECUC_02148]	Replacement for EcucSymbolicNameReferenceDef
[TPS_ECUC_06085]	Ordering of MetaDataItems of an MetaDataType

Table D.44: Deleted Specification Items in 19-11

D.12.4 Added Constraints in 19-11

Number	Heading
[constr_5059]	Ordering of MetaDataItems of a MetaDataType

Table D.45: Added Constraints in 19-11

D.12.5 Changed Constraints in 19-11

none

D.12.6 Deleted Constraints in 19-11

none

D.13 Change History for release R20-11

D.13.1 Added Specification Items in R20-11

Number	Heading
[TPS_ECUC_02149]	Existence of EcucDefinitionCollection.module
[TPS_ECUC_02150]	Existence of EcucModuleDef.container
[TPS_ECUC_02151]	Existence of EcucValueCollection.ecucValue
[TPS_ECUC_02152]	EcucModuleConfigurationValues.container

Table D.46: Added Specification Items in R20-11

D.13.2 Changed Specification Items in R20-11

none

D.13.3 Deleted Specification Items in R20-11

Number	Heading
[TPS_ECUC_02088]	Configuration Editor shall display the content of the longName to users
[TPS_ECUC_04002]	ECU Configuration Editor shall be able to merge ECU Configuration Value descriptions
[TPS_ECUC_04003]	ECU Configuration Editor shall be able to work with subsets of parameters
[TPS_ECUC_04005]	ECU Configuration Editor shall be able to generate and import EcucModule-ConfigurationValues
[TPS_ECUC_06071]	ECU Configuration Editor shall be able to read parameter values in any order
[TPS_ECUC_06073]	The ECU Configuration Editor shall be able to work with arbitrary package structures

Table D.47: Deleted Specification Items in R20-11

D.13.4 Added Constraints in R20-11

Number	Heading
[constr_3570]	EcucDefinitionElement.lowerMultiplicity always required
[constr_3571]	EcucCommonAttributes.origin always required
[constr_3572]	EcucParameterDef.symbolicNameValue always required
[constr_3573]	EcucAbstractConfigurationClass.configClass always required
[constr_3574]	EcucAbstractConfigurationClass.configVariant always required
[constr_3575]	EcucEnumerationLiteralDef.origin always required
[constr_3576]	EcucInstanceReferenceDef.destinationContext always required
[constr_3577]	EcucInstanceReferenceDef.destinationType always required
[constr_3578]	EcucForeignReferenceDef.destinationType always required
[constr_3579]	EcucReferenceDef.destination always required
[constr_3580]	EcucUriReferenceDef.destinationUri always required
[constr_3581]	EcucDestinationUriDefSet.destinationUriDef always required
[constr_3582]	EcucDestinationUriDef.destinationUriPolicy always required
[constr_3583]	EcucDestinationUriPolicy.destinationUriNestingContract always required
[constr_3584]	EcucQuery.ecucQueryExpression always required
[constr_3585]	EcucConditionFormula.ecucQuery always required
[constr_3586]	EcucConditionFormula.ecucQueryString always required
[constr_3587]	EcucValidationCondition.validationFormula always required
[constr_3588]	EcucValueCollection.ecuExtract always required
[constr_3589]	EcucModuleConfigurationValues.ecucDefEdition always required
[constr_3590]	EcucModuleConfigurationValues.implementationConfigVariant always required
[constr_3591]	EcucModuleConfigurationValues.definition always required
[constr_3592]	EcucContainerValue.definition always required
[constr_3593]	EcucParameterValue.definition always required
[constr_3594]	EcucNumericalParamValue.value always required
[constr_3595]	EcucTextualParamValue.value always required
[constr_3596]	EcucAddInfoParamValue.value always required
[constr_3597]	EcucAbstractReferenceValue.definition always required
[constr_3598]	EcucInstanceReferenceValue.value always required
[constr_3599]	EcucReferenceValue.value always required
[constr_5108]	CddModuleId range restriction

Table D.48: Added Constraints in R20-11

D.13.5 Changed Constraints in R20-11

none

D.13.6 Deleted Constraints in R20-11

none

D.14 Change History for release R21-11

D.14.1 Added Specification Items in R21-11

none

D.14.2 Changed Specification Items in R21-11

Number	Heading
[TPS_ECUC_06069]	Sorting criteria for Parameters on the Values side

Table D.49: Changed Specification Items in R21-11

D.14.3 Deleted Specification Items in R21-11

Number	Heading
[TPS_ECUC_02006]	Container definition
[TPS_ECUC_02043]	Each `EcucContainerDef` is `Identifiable`

Table D.50: Deleted Specification Items in R21-11

D.14.4 Added Constraints in R21-11

none

D.14.5 Changed Constraints in R21-11

Number	Heading
[constr_5520]	`valueConfigClass` attribute of `symbolicNameValue` parameters

Table D.51: Changed Constraints in R21-11

D.14.6 Deleted Constraints in R21-11

none

D.15 Change History for release R22-11

D.15.1 Added Specification Items in R22-11

Number	Heading
[TPS_ECUC_06089]	Multiple aggregation of container trees that include references to other subContainers in the same aggregated container tree

Table D.52: Added Specification Items in R22-11

D.15.2 Changed Specification Items in R22-11

Number	Heading
[TPS_ECUC_02009]	Expression of optionality of containers, parameters and references in the Ecuc Parameter Definition UML model
[TPS_ECUC_02108]	Rule for the creation of #define symbols in the header file for parameters with the symbolicNameValue set to TRUE
[TPS_ECUC_06037]	apiServicePrefix attribute for Complex Driver module and Xfrm module

Table D.53: Changed Specification Items in R22-11

D.15.3 Deleted Specification Items in R22-11

Number	Heading
[TPS_ECUC_06017]	Existence of upperMultiplicityInfinite and upperMultiplicity is mutually exclusive

Table D.54: Deleted Specification Items in R22-11

D.15.4 Added Constraints in R22-11

Number	Heading
[constr_5325]	Existence of upperMultiplicityInfinite and upperMultiplicity is mutually exclusive
[constr_5342]	EcucDefinitionElement.upperMultiplicity or EcucDefinitionElement.upperMultiplicityInfinite always required
[constr_5345]	Restriction for a reference destination in case of multiple aggregated EcucParamConfContainerDefs

Table D.55: Added Constraints in R22-11

D.15.5 Changed Constraints in R22-11

Number	Heading
[constr_3023]	Usage of `apiServicePrefix`

Table D.56: Changed Constraints in R22-11

D.15.6 Deleted Constraints in R22-11

none

D.16 Change History for release R23-11

D.16.1 Added Specification Items in R23-11

Number	Heading
[TPS_ECUC_06091]	Matrix of allowed status value combinations of EcucParamConfContainerDef and aggregations of EcucParameterDef/EcucAbstractReferenceDef/EcucContainerDef in the StMD
[TPS_ECUC_06092]	Matrix of allowed status value combinations of referenced targets of a `EcucAbstractReferenceDef`
[TPS_ECUC_06093]	J1939Rm callback functions

Table D.57: Added Specification Items in R23-11

D.16.2 Changed Specification Items in R23-11

Number	Heading
[TPS_ECUC_02042]	Specification of the `destinationType` in a `EcucForeignReferenceDef`
[TPS_ECUC_02061]	Specification of the destinationType in a EcucInstanceReferenceDef
[TPS_ECUC_02082]	Specification of the `destinationType` in a `EcucInstanceReferenceDef`
[TPS_ECUC_06082]	Definition of interval type for `EcucFloatParamDef.min` and `EcucFloatParamDef.max`
[TPS_ECUC_06088]	Specification of the `destinationType` format in a `EcucForeignReferenceDef`

Table D.58: Changed Specification Items in R23-11

D.16.3 Deleted Specification Items in R23-11

Number	Heading
[TPS_ECUC_02015]	Origin information in parameter and reference definitions

Table D.59: Deleted Specification Items in R23-11

D.16.4 Added Constraints in R23-11

Number	Heading
[constr_5365]	Origin information in parameter and reference definitions

Table D.60: Added Constraints in R23-11

D.16.5 Changed Constraints in R23-11

none

D.16.6 Deleted Constraints in R23-11

none

D.17 Change History for release R24-11

D.17.1 Added Specification Items in R24-11

Number	Heading
[ECUC_EcuC_00088]	Definition of EcucIntegerParamDef PduBufferAlignment

Table D.61: Added Specification Items in R24-11

D.17.2 Changed Specification Items in R24-11

Number	Heading
[ECUC_EcuC_00001]	Definition of EcucParamConfContainerDef Pdu
[ECUC_EcuC_00005]	Definition of EcucParamConfContainerDef EcucPartition
[ECUC_EcuC_00076]	Definition of EcucEnumerationParamDef MetaDataItemType

▽

△

Number	Heading
[ECUC_EcuC_00084]	Definition of EcucDefinitionCollection AUTOSARParameterDefinition
[TPS_ECUC_03037]	Deriving the symbolic name in the implementation from the `shortName` of the referenced container

Table D.62: Changed Specification Items in R24-11

D.17.3 Deleted Specification Items in R24-11

Number	Heading
[ECUC_EcuC_00006]	Definition of EcucBooleanParamDef PartitionCanBeRestarted
[ECUC_EcuC_00037]	Definition of EcucBooleanParamDef EcucDefaultBswPartition

Table D.63: Deleted Specification Items in R24-11

D.17.4 Added Constraints in R24-11

Number	Heading
[constr_3793]	Usage of `KeepLocalPduBuffer`
[constr_3794]	Usage of `PduBufferAlignment`

Table D.64: Added Constraints in R24-11

D.17.5 Changed Constraints in R24-11

none

D.17.6 Deleted Constraints in R24-11

none

D.18 Change History for release R25-11

D.18.1 Added Specification Items in R25-11

Number	Heading
[ECUC_EcuC_00089]	Definition of EcucParamConfContainerDef EcucBusToPartitionMappingSet
[ECUC_EcuC_00090]	Definition of EcucParamConfContainerDef EcucBusToPartitionMapping
[ECUC_EcuC_00091]	Definition of EcucReferenceDef EcucBusToPartitionMappingBusRef
[ECUC_EcuC_00092]	Definition of EcucReferenceDef EcucBusToPartitionMappingPartitionRef
[ECUC_EcuC_00093]	Definition of EcucForeignReferenceDef EcucBusToPartitionMappingPhysicalChannelRef
[TPS_ECUC_01050]	Semantics of `EcucUriReferenceDef`
[TPS_ECUC_01051]	Constraints for the structure of the target of an URI-based reference
[TPS_ECUC_01052]	`EcucDestinationUriPolicy.destinationUriNestingContract` is set to `targetContainer`
[TPS_ECUC_01053]	`EcucDestinationUriPolicy.destinationUriNestingContract` is set to `leafOfTargetContainer`
[TPS_ECUC_01054]	`EcucDestinationUriPolicy.destinationUriNestingContract` is set to `vertexOfTargetContainer`

Table D.65: Added Specification Items in R25-11

D.18.2 Changed Specification Items in R25-11

Number	Heading
[ECUC_EcuC_00008]	Definition of EcucModuleDef EcuC
[ECUC_EcuC_00076]	Definition of EcucEnumerationParamDef MetaDataItemType
[ECUC_EcuC_00084]	Definition of EcucDefinitionCollection AUTOSARParameterDefinition
[TPS_ECUC_02033]	`EcucFunctionNameDef` properties

Table D.66: Changed Specification Items in R25-11

D.18.3 Deleted Specification Items in R25-11

Number	Heading
[ECUC_Mcu_00174]	Definition of EcucParamConfContainerDef McuClockReferencePoint
[ECUC_Mcu_00175]	Definition of EcucFloatParamDef McuClockReferencePointFrequency
[TPS_ECUC_06066]	Order of Container-, Parameter- and Reference-Values
[TPS_ECUC_06067]	Sorting criteria for Containers on the Values side
[TPS_ECUC_06068]	Sorting criteria for References on the Values side

▽

△

Number	Heading
[TPS_ECUC_06069]	Sorting criteria for Parameters on the Values side
[TPS_ECUC_06078]	`EcucUriReferenceDef` properties
[TPS_ECUC_06079]	`EcucDestinationUriPolicy.destinationUriNestingContract` is set to `targetContainer`
[TPS_ECUC_06080]	`EcucDestinationUriPolicy.destinationUriNestingContract` is set to `leafOfTargetContainer`
[TPS_ECUC_06081]	`EcucDestinationUriPolicy.destinationUriNestingContract` is set to `vertexOfTargetContainer`

Table D.67: Deleted Specification Items in R25-11

D.18.4 Added Constraints in R25-11

Number	Heading
[constr_13000]	Existence of reference in the role `EcucContainerDef.destinationUri` for an `EcucContainerDef` aggregated in the role `EcucDestinationUriPolicy.container`

Table D.68: Added Constraints in R25-11

D.18.5 Changed Constraints in R25-11

Number	Heading
[constr_3120]	Applicable attributes when `EcucDestinationUriPolicy.destinationUriNestingContract` is set to `targetContainer`
[constr_5521]	`multiplicityConfigClass` attribute of `symbolicNameValue` parameters
[constr_5522]	`postBuildVariantMultiplicity` attribute of `symbolicNameValue` parameters

Table D.69: Changed Constraints in R25-11

D.18.6 Deleted Constraints in R25-11

Number	Heading
[constr_3119]	Necessary content of `EcucDestinationUriDef`s that are referenced by an `EcucContainerDef`

▽

△

Number	Heading
[constr_3509]	Applicability of `scope` attribute

Table D.70: Deleted Constraints in R25-11

D.19 Change History for release R26-11

D.19.1 Added Specification Items in R26-11

Number	Heading
[ECUC_EcuC_00094]	Definition of EcucIntegerParamDef EcucDmaThreshold
[TPS_ECUC_01059]	`EcucModuleDef` that relies on `EcucCommonAttributes` with `postBuildVariantValue` set to true of another `EcucModuleDef`
[TPS_ECUC_01060]	`EcucModuleDef` that relies on `EcucCommonAttributes` with `postBuildVariantMultiplicity` set to true of another `EcucModuleDef`
[TPS_ECUC_01063]	Impact of `postBuildVariantUsed` value set to FALSE
[TPS_ECUC_01064]	Introduction of new `EcucParameterValue`s of type `EcucFunctionNameDef` at post-build time
[TPS_ECUC_06094]	Module support for post-build loadable
[TPS_ECUC_06095]	Semantics of `postBuildLoadableMultiplicity` attribute
[TPS_ECUC_06096]	Latest configuration time to change the number of instances of `EcucContainerDef`s according to the StMD
[TPS_ECUC_06097]	Latest configuration time to change the number of instances of `EcucContainerDef`s according to the VSMD
[TPS_ECUC_06098]	Latest configuration time to change the number of instances of parameters or references according to the StMD
[TPS_ECUC_06099]	Latest configuration time to change the value of parameters or references according to the StMD
[TPS_ECUC_06100]	Latest configuration time to change the number of instances of parameters or references according to the VSMD
[TPS_ECUC_06101]	Latest configuration time to change the value of parameters or references according to the VSMD
[TPS_ECUC_06102]	Semantics of `EcucCommonAttributes.postBuildLoadableMultiplicity` attribute
[TPS_ECUC_06103]	Semantics of `EcucCommonAttributes.postBuildLoadableValue` attribute
[TPS_ECUC_06104]	`EcucModuleDef` that relies on `EcucCommonAttributes` with `postBuildLoadableValue` = true of another `EcucModuleDef`
[TPS_ECUC_06105]	`EcucModuleDef` that relies on `EcucCommonAttributes` with `postBuildVariantValue` = true of another `EcucModuleDef`
[TPS_ECUC_06106]	`EcucModuleDef` that relies on `EcucCommonAttributes` with `postBuildLoadableMultiplicity` = true of another `EcucModuleDef`

▽

△

Number	Heading
[TPS_ECUC_06107]	EcucModuleDef that relies on EcucCommonAttributes with postBuildVariantMultiplicity = true of another EcucModuleDef
[TPS_ECUC_06108]	EcucModuleDef that relies on EcucCommonAttributes with linkTimeSupport = true of another EcucModuleDef
[TPS_ECUC_06109]	The value of the EcucModuleDef.postBuildLoadableSupport attribute in the VSMD in case it is not defined in the StMD
[TPS_ECUC_06110]	The value of the EcucModuleDef.postBuildLoadableSupport attribute in the VSMD in case it is set to false in the StMD
[TPS_ECUC_06111]	The value of the EcucModuleDef.postBuildLoadableSupport attribute in the VSMD in case it is set to true in the StMD
[TPS_ECUC_06112]	The value of the EcucContainerDef.postBuildLoadableMultiplicity attribute in the VSMD in case it is set to false in the StMD
[TPS_ECUC_06113]	The value of the EcucContainerDef.postBuildLoadableMultiplicity in the VSMD in case it is set to true in the StMD
[TPS_ECUC_06114]	The value of the EcucParameterDef.postBuildLoadableValue and the EcucAbstractReferenceDef.postBuildLoadableValue attributes in the VSMD in case they are set to false in the StMD
[TPS_ECUC_06115]	The value of the EcucParameterDef.postBuildLoadableMultiplicity and the EcucAbstractReferenceDef.postBuildLoadableMultiplicity attributes in the VSMD in case they are set to false in the StMD
[TPS_ECUC_06116]	The value of the EcucParameterDef.postBuildLoadableValue and the EcucAbstractReferenceDef.postBuildLoadableValue attributes in the VSMD in case they are set to true in the StMD
[TPS_ECUC_06117]	The value of the EcucParameterDef.postBuildLoadableMultiplicity and the EcucAbstractReferenceDef.postBuildLoadableMultiplicity attributes in the VSMD in case they are set to true in the StMD
[TPS_ECUC_06118]	EcucModuleDef that relies on EcucContainerDefs with postBuildVariantMultiplicity set to true of another EcucModuleDef
[TPS_ECUC_06119]	EcucModuleDef that relies on EcucContainerDefs with postBuildLoadableMultiplicity set to true of another EcucModuleDef
[TPS_ECUC_06120]	Usage of postBuildLoadableMultiplicity attribute is independent of its aggregated subContainers

Table D.71: Added Specification Items in R26-11

D.19.2 Changed Specification Items in R26-11

Number	Heading
[ECUC_EcuC_00002]	Definition of EcucParamConfContainerDef EcucPduCollection
[ECUC_EcuC_00052]	Definition of EcucForeignReferenceDef SysTPduToFrameTriggeringRef
[ECUC_EcuC_00054]	Definition of EcucForeignReferenceDef SysTPduToPduTriggeringRef
[ECUC_EcuC_00057]	Definition of EcucParamConfContainerDef EcucCoreDefinition
[ECUC_EcuC_00084]	Definition of EcucDefinitionCollection AUTOSARParameterDefinition
[TPS_ECUC_01052]	Target type implements `EcucDestinationUriPolicy.container`
[TPS_ECUC_02139]	Definition of precompile/linktime/postbuild support for all CDD configuration parameters and references
[TPS_ECUC_08002]	Introduction of new `EcucParamConfContainerDef` instances in post-build variants
[TPS_ECUC_08012]	Module support for post-build variants
[TPS_ECUC_08013]	Different number of `EcucContainerDef` instances in different post-build selectable variants
[TPS_ECUC_08015]	Different number of `EcucCommonAttributes`s instances in post-build selectable variants
[TPS_ECUC_08043]	The number of `EcucContainerValue` instances in post-build loadable variants
[TPS_ECUC_08045]	The value of `EcucParameterValue` instances in post-build loadable variants
[TPS_ECUC_08047]	The number of `EcucParameterValue` instances in post-build loadable variants
[TPS_ECUC_08049]	The value of `EcucAbstractReferenceValue` instances in post-build loadable variants
[TPS_ECUC_08051]	The number of `EcucAbstractReferenceValue` instances in post-build loadable variants

Table D.72: Changed Specification Items in R26-11

D.19.3 Deleted Specification Items in R26-11

Number	Heading
[ECUC_EcuC_00041]	Definition of EcucEnumerationParamDef PduIdTypeEnum
[ECUC_EcuC_00042]	Definition of EcucEnumerationParamDef PduLengthTypeEnum
[TPS_ECUC_01025]	Generate and extract activities are fully automatic
[TPS_ECUC_01051]	Constraints for the structure of the target of an URI-based reference
[TPS_ECUC_01053]	`EcucDestinationUriPolicy.destinationUriNestingContract` is set to `leafOfTargetContainer`
[TPS_ECUC_01054]	`EcucDestinationUriPolicy.destinationUriNestingContract` is set to `vertexOfTargetContainer`
[TPS_ECUC_02016]	Configuration class of parameter and reference definitions

▽

△

Number	Heading
[TPS_ECUC_02017]	Configuration class "PreCompile"
[TPS_ECUC_02018]	Configuration class "Link"
[TPS_ECUC_02019]	Configuration class "PostBuild"
[TPS_ECUC_02070]	Configuration class "PubilishedInformation"
[TPS_ECUC_02071]	Usage of PublishedInformation configuration class
[TPS_ECUC_02096]	Supported configuration variants in a BSW module
[TPS_ECUC_02097]	Supported configuration variants in the StMD and the VSMD
[TPS_ECUC_02098]	StMD Configuration variant "VariantPreCompile"
[TPS_ECUC_02099]	StMD Configuration variant "VariantLinkTime"
[TPS_ECUC_02100]	StMD Configuration variant "VariantPostBuild"
[TPS_ECUC_02101]	EcucAbstractConfigurationClass usage
[TPS_ECUC_02102]	Configuration class selection for parameters and references for supported configuration variants
[TPS_ECUC_02103]	Configuration variant of EcucModuleConfigurationValues
[TPS_ECUC_02124]	Tooling approaches that are supported by the EcuC Parameter Definition and ECU Configuration Values
[TPS_ECUC_02144]	Definition of supported config variants for CDD
[TPS_ECUC_02149]	Existence of EcucDefinitionCollection.module
[TPS_ECUC_02150]	Existence of EcucModuleDef.container
[TPS_ECUC_02151]	Existence of EcucValueCollection.ecucValue
[TPS_ECUC_02152]	EcucModuleConfigurationValues.container
[TPS_ECUC_08000]	Different number of EcucContainerDef instances in different configuration times
[TPS_ECUC_08003]	Usage of multiplicityConfigClass.configClass attribute is independent of its aggregated subContainers
[TPS_ECUC_08005]	The value of the EcucContainerDef.multiplicityConfigClass attribute in the VSMD in case it is not defined in the StMD
[TPS_ECUC_08006]	The value of the EcucContainerDef.multiplicityConfigClass attribute in the VSMD in case it is defined in the StMD
[TPS_ECUC_08025]	The value of the EcucContainerDef.postBuildVariantMultiplicity attribute in the VSMD in case it is not defined in the StMD
[TPS_ECUC_08028]	The value of the EcucParameterDef.postBuildVariantMultiplicity and the EcucAbstractReferenceDef.postBuildVariantMultiplicity attributes in the VSMD in case they are not defined in the StMD
[TPS_ECUC_08029]	The value of the EcucParameterDef.postBuildVariantValue and the EcucAbstractReferenceDef.postBuildVariantValue attributes in the VSMD in case they are not defined in the StMD
[TPS_ECUC_08034]	Different values of EcucCommonAttributes instances in different configuration times

▽

△

Number	Heading
[TPS_ECUC_08035]	Different number of instances of EcucCommonAttributess in different configuration times
[TPS_ECUC_08036]	The value of the EcucParameterDef.valueConfigClass and the EcucAbstractReferenceDef.valueConfigClass attributes in the VSMD in case they are not defined in the StMD
[TPS_ECUC_08037]	The value of the EcucParameterDef.multiplicityConfigClass and the EcucAbstractReferenceDef.multiplicityConfigClass attributes in the VSMD in case they are not defined in the StMD
[TPS_ECUC_08038]	The value of the EcucParameterDef.valueConfigClass and the EcucAbstractReferenceDef.valueConfigClass attributes in the VSMD in case they are defined in the StMD
[TPS_ECUC_08039]	The value of the EcucParameterDef.multiplicityConfigClass and the EcucAbstractReferenceDef.multiplicityConfigClass attributes in the VSMD in case they are defined in the StMD

Table D.73: Deleted Specification Items in R26-11

D.19.4 Added Constraints in R26-11

Number	Heading
[constr_4200]	Existence of EcucDefinitionCollection.module
[constr_4201]	Existence of EcucModuleDef.container
[constr_4202]	Existence of EcucValueCollection.ecucValue
[constr_4203]	EcucModuleConfigurationValues.container
[constr_9439]	Existence of attribute EcucModuleDef.postBuildVariantSupport
[constr_9440]	Existence of attribute EcucModuleDef.postBuildLoadableSupport
[constr_9441]	Value of EcucContainerDef.postBuildLoadableMultiplicity if postBuildLoadableSupport is set to false
[constr_9442]	Value of postBuildLoadableMultiplicity if postBuildLoadableSupport is set to false
[constr_9443]	Value of postBuildLoadableValue if postBuildLoadableSupport is set to false
[constr_9444]	Mutual exclusivity of linkTimeSupport and postBuildLoadableMultiplicity
[constr_9445]	EcucContainerDef.linkTimeSupport not applicable in StMD
[constr_9446]	Existence of the postBuildLoadableMultiplicity attribute
[constr_9447]	Removing an instance of the EcucContainerDef at post-build time
[constr_9450]	EcucCommonAttributes.linkTimeSupport not applicable in StMD
[constr_9451]	Existence of attribute EcucCommonAttributes.linkTimeSupport
[constr_9452]	Mutual exclusivity of linkTimeSupport and postBuildLoadableMultiplicity

▽

△

Number	Heading
[constr_9453]	Mutual exclusivity of `linkTimeSupport` and `postBuildLoadableValue`
[constr_9454]	Existence of attribute `EcucCommonAttributes.postBuildVariantValue`
[constr_9455]	Existence of attribute `EcucCommonAttributes.postBuildVariantMultiplicity`
[constr_9456]	Existence of attribute `EcucCommonAttributes.postBuildLoadableValue`
[constr_9457]	Existence of attribute `EcucCommonAttributes.postBuildLoadableMultiplicity`
[constr_9458]	`EcucModuleDef` that relies on `EcucCommonAttributes` with `postBuildLoadableValue` set to true of another `EcucModuleDef`
[constr_9459]	`EcucModuleDef` that relies on `EcucCommonAttributes` with `postBuildLoadableMultiplicity` set to true of another `EcucModuleDef`

Table D.74: Added Constraints in R26-11

D.19.5 Changed Constraints in R26-11

Number	Heading
[constr_3120]	`EcucDestinationUriPolicy.container` always required
[constr_3794]	Usage of `PduBufferAlignment`
[constr_5506]	Existence of `postBuildVariantMultiplicity` attribute
[constr_5520]	Latest configuration time to change the values of `symbolicNameValue` parameters
[constr_5521]	Latest configuration time to change the number of instances of `symbolicNameValue` parameters

Table D.75: Changed Constraints in R26-11

D.19.6 Deleted Constraints in R26-11

Number	Heading
[constr_3091]	Multiplicity of `valueConfigClass`
[constr_3092]	Usage of `configVariant` and `configClass` attributes
[constr_3233]	`EcucModuleDef` that relies on `EcucCommonAttributes` with `valueConfigClass` set to `Link`/`PostBuild` of another `EcucModuleDef`
[constr_3234]	`EcucModuleDef` that relies on `EcucCommonAttributes` with `multiplicityConfigClass` set to `Link`/`PostBuild` of another `EcucModuleDef`

▽

△

Number	Heading
[constr_3235]	EcucModuleDef that relies on EcucContainerDefs with multiplicityConfigClass set to Link/PostBuild of another EcucModuleDef
[constr_3236]	EcucModuleDef that relies on EcucCommonAttributes with postBuildVariantValue set to true of another EcucModuleDef
[constr_3237]	EcucModuleDef that relies on EcucCommonAttributes with postBuildVariantMultiplicity set to true of another EcucModuleDef
[constr_3238]	EcucModuleDef that relies on EcucContainerDef with postBuildVariantMultiplicity set to true of another EcucModuleDef
[constr_3449]	Impact of postBuildVariantUsed value set to FALSE
[constr_3573]	EcucAbstractConfigurationClass.configClass always required
[constr_3574]	EcucAbstractConfigurationClass.configVariant always required
[constr_3583]	EcucDestinationUriPolicy.destinationUriNestingContract always required
[constr_3590]	EcucModuleConfigurationValues.implementationConfigVariant always required
[constr_5015]	Multiplicity of multiplicityConfigClass
[constr_5500]	Applicability of the multiplicityConfigClass attribute
[constr_5502]	Introduction of new EcucParameterValues of type EcucFunctionNameDef at post-build time
[constr_5504]	Removing an instance of the EcucContainerDef at post-build time
[constr_5508]	Applicability of postBuildVariantMultiplicity attribute
[constr_5512]	postBuildVariantValue attribute of symbolicNameValue parameters
[constr_5514]	Applicability of the multiplicityConfigClass attribute
[constr_5522]	postBuildVariantMultiplicity attribute of symbolicNameValue parameters
[constr_5523]	Allowed configClasses for paired configVariants

Table D.76: Deleted Constraints in R26-11

E 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	***ARElement*** (abstract)			
Note	An element that can be defined stand-alone, i.e. without being part of another element (except for packages of course).			
Base	*ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	*AbstractMessageToSecureChannelMapping*, AclObjectSet, AclOperation, AclPermission, AclRole, AliasNameSet, ApplicabilityInfoSet, ApplicationPartition, *AutosarDataType*, *BaseType*, BlueprintMappingSet, BswEntryRelationshipSet, BswModuleDescription, BswModuleEntry, BuildActionManifest, CalibrationParameterValueSet, ClientIdDefinitionSet, Collection, CompuMethod, ConsistencyNeedsBlueprintSet, ConstantSpecification, ConstantSpecificationMappingSet, CpSoftwareCluster, CpSoftwareClusterBinaryManifestDescriptor, CpSoftwareClusterMappingSet, CpSoftwareClusterResourcePool, CryptoEllipticCurveProps, CryptoServiceCertificate, CryptoServiceKey, CryptoServicePrimitive, CryptoServiceQueue, CryptoSignatureScheme, DataConstr, DataTransformationSet, DataTypeMappingSet, DdsCpConfig, *DiagnosticCommonElement*, DiagnosticConnection, DiagnosticContributionSet, DltArgumentPropsSet, DltContext, DltEcu, Documentation, E2EProfileCompatibilityProps, EcucDefinitionCollection, EcucDestinationUriDefSet, EcucModuleConfigurationValues, EcucModuleDef, EcucValueCollection, EthIpProps, EthTcpIpIcmpProps, EthTcpIpProps, EvaluatedVariantSet, FMFeature, FMFeatureMap, FMFeatureModel, FMFeatureSelectionSet, FirewallRule, FlatMap, GeneralPurposeConnection, HwCategory, HwElement, HwType, *IEEE1722TpConnection*, IPSecConfigProps, IPv6ExtHeaderFilterSet, *IdsCommonElement*, IdsDesign, *Implementation*, ImpositionTimeDefinitionGroup, InterpolationRoutineMappingSet, J1939ControllerApplication, KeywordSet, LifeCycleInfoSet, LifeCycleStateDefinitionGroup, LogAndTraceMessageCollectionSet, MacSecGlobalKayProps, MacSecParticipantSet, McFunction, McGroup, ModeDeclarationGroup, ModeDeclarationMappingSet, OsTaskProxy, PhysicalDimension, PhysicalDimensionMappingSet, *PortInterface*, PortInterfaceMappingSet, PortPrototypeBlueprint, PostBuildVariantCriterion, PostBuildVariantCriterionValueSet, PredefinedVariant, RapidPrototypingScenario, SdgDef, *SecureComProps*, *SecurityConnectivityAssociation*, SignalServiceTranslationPropsSet, SomeipSdClientEventGroupTimingConfig, SomeipSdClientServiceInstanceConfig, SomeipSdServerEventGroupTimingConfig, SomeipSdServerServiceInstanceConfig, SwAddrMethod, SwAxisType, SwComponentMappingConstraints, *SwComponentType*, SwRecordLayout, SwSystemconst, SwSystemconstantValueSet, SwcBswMapping, System, SystemComSpecDefinitionSet, SystemSignal, SystemSignalGroup, TDCpSoftwareClusterMappingSet, TcpOptionFilterSet, *TimingExtension*, TlsConnectionGroup, TlvDataIdDefinitionSet, TransformationPropsSet, Unit, UnitGroup, *UploadablePackageElement*, ViewMapSet			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table E.1: ARElement

Class	**ARPackage**			
Note	AUTOSAR package, allowing to create top level packages to structure the contained ARElements. ARPackages are open sets. This means that in a file based description system multiple files can be used to partially describe the contents of a package. This is an extended version of MSR's SW-SYSTEM.			
Base	*ARObject*, *AtpBlueprint*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	ARPackage.arPackage, AUTOSAR.arPackage			
Attribute	***Type***	***Mult.***	***Kind***	***Note***



△

Class	**ARPackage**			
arPackage	ARPackage	*	aggr	This represents a sub package within an ARPackage, thus allowing for an unlimited package hierarchy. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=arPackage, arPackage.shortName, ar Package.variationPoint.shortLabel vh.latestBindingTime=blueprintDerivationTime xml.sequenceOffset=30
element	PackageableElement	*	aggr	Elements that are part of this package **Stereotypes:** atpVariation **Tags:** atp.Splitkey=element, element.shortName, element.variationPoint.shortLabel vh.latestBindingTime=systemDesignTime xml.sequenceOffset=20
referenceBase	ReferenceBase	*	aggr	This denotes the reference bases for the package. This is the basis for all relative references within the package. The base needs to be selected according to the base attribute within the references. **Tags:** atp.Splitkey=referenceBase, referenceBase.shortLabel xml.sequenceOffset=10

Table E.2: ARPackage

Class	**AUTOSAR**			
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:** atp.Splitkey=adminData xml.sequenceOffset=10
arPackage	ARPackage	*	aggr	This is the top level package in an AUTOSAR model. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=arPackage, arPackage.shortName, ar Package.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:** atp.Splitkey=fileInfoComment xml.roleElement=true xml.sequenceOffset=-10 xml.typeElement=false

▽

△

Class	**AUTOSAR**			
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:** atp.Splitkey=introduction xml.sequenceOffset=20

Table E.3: AUTOSAR

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

▽

△

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

Table E.4: AdminData

Class	**AnyInstanceRef**			
Note	Describes a reference to any instance in an AUTOSAR model. This is the most generic form of an instance ref. Refer to the superclass notes for more details.			
Base	*ARObject*, *AtpInstanceRef*			
Aggregated by	ApmcInstanceReferenceValue.value, ApmcUpstreamDocInstanceReferenceValue.value, ApmcUri InstanceReferenceValue.value, Collection.collectedInstance, Collection.sourceInstance, Documentation Context.feature, EcucInstanceReferenceValue.value, FlatInstanceDescriptor.ecuExtractReference, Flat InstanceDescriptor.upstreamReference, RptContainer.byPassPoint, RptHook.rptArHook, SecurityEvent ReportInstanceValue.object, ViewMap.firstElementInstance, ViewMap.secondElementInstance			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
base	AtpClassifier	0..1	ref	This is the base from which navigation path begins. **Stereotypes:** atpDerived
contextElement (ordered)	AtpPrototype	*	ref	This is one step in the navigation path specified by the instance ref.
target	AtpFeature	0..1	ref	This is the target of the instance ref.

Table E.5: AnyInstanceRef

Class	**AssemblySwConnector**			
Note	`AssemblySwConnector`s are exclusively used to connect `SwComponentPrototype`s in the context of a `CompositionSwComponentType`.			
Base	*ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *SwConnector*			
Aggregated by	*AtpClassifier*.atpFeature, CompositionSwComponentType.connector			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
provider	AbstractProvidedPort Prototype	0..1	iref	Instance of providing port. **Tags:** atp.Splitkey=provider, provider.contextComponent, provider.targetPPort **InstanceRef implemented by:** PPortInComposition InstanceRef
requester	AbstractRequiredPort Prototype	0..1	iref	Instance of requiring port. **Tags:** atp.Splitkey=requester, requester.context Component, requester.targetRPort **InstanceRef implemented by:** RPortInComposition InstanceRef

Table E.6: AssemblySwConnector

Class	**BswImplementation**			
Note	Contains the implementation specific information in addition to the generic specification (BswModule Description and BswBehavior). It is possible to have several different BswImplementations referring to the same BswBehavior. **Tags:** atp.recommendedPackage=BswImplementations This Class is only used by the AUTOSAR Classic Platform.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *Implementation*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
arRelease Version	RevisionLabelString	0..1	attr	Version of the AUTOSAR Release on which this implementation is based. The numbering contains three levels (major, minor, revision) which are defined by AUTOSAR.
behavior	BswInternalBehavior	0..1	ref	The behavior of this implementation. This relation is made as an association because • it follows the pattern of the SWCT • since ARElement cannot be split, but we want supply the implementation later, the BswImplementation is not aggregated in BswBehavior **Tags:** atp.Splitkey=behavior
preconfigured Configuration	EcucModule ConfigurationValues	*	ref	Reference to the set of preconfigured (i.e. fixed) configuration values for this BswImplementation. If the BswImplementation represents a cluster of several modules, more than one EcucModuleConfigurationValues element can be referred (at most one per module), otherwise at most one such element can be referred. **Tags:** atp.Splitkey=preconfiguredConfiguration xml.roleWrapperElement=true
recommended Configuration	EcucModule ConfigurationValues	*	ref	Reference to one or more sets of recommended configuration values for this module or module cluster. **Tags:** atp.Splitkey=recommendedConfiguration
vendorApiInfix	Identifier	0..1	attr	In driver modules which can be instantiated several times on a single ECU, SRS_BSW_00347 requires that the names of files, APIs, published parameters and memory allocation keywords are extended by the vendorId and a vendor specific name. This parameter is used to specify the vendor specific name. In total, the implementation specific API name is generated as follows: <Module Name>_<vendorId>_ <vendorApiInfix>_<API name from SWS>. E.g. assuming that the vendorId of the implementer is 123 and the implementer chose a vendorApiInfix of "v11r456" an API name Can_Write defined in the SWS will translate to Can_123_v11r456_Write. This attribute is mandatory for all modules with upper multiplicity > 1. It shall not be used for modules with upper multiplicity =1. See also SWS_BSW_00102.

▽

△

Class	**BswImplementation**			
vendorSpecific ModuleDef	EcucModuleDef	*	ref	Reference to • the vendor specific EcucModuleDef used in this Bsw Implementation if it represents a single module • several EcucModuleDefs used in this Bsw Implementation if it represents a cluster of modules • one or no EcucModuleDefs used in this Bsw Implementation if it represents a library **Tags:** atp.Splitkey=vendorSpecificModuleDef xml.roleWrapperElement=true

Table E.7: BswImplementation

Primitive	**CIdentifier**			
Note	This datatype represents a string, that follows the rules of C-identifiers. **Tags:** xml.xsd.customType=C-IDENTIFIER xml.xsd.pattern=[a-zA-Z_][a-zA-Z0-9_]* xml.xsd.type=string			
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
namePattern	String	0..1	attr	This attribute represents a pattern which shall be used to define the value of the identifier if the CIdentifier in question is part of a blueprint. For more details refer to TPS_StandardizationTemplate. **Tags:** xml.attribute=true

Table E.8: CIdentifier

Class	**CompositionSwComponentType**			
Note	A `CompositionSwComponentType` aggregates `SwComponentPrototype`s (that in turn are typed by `SwComponentType`)s as well as `SwConnector`s for primarily connecting `SwComponentPrototype`s among each others and towards the surface of the `CompositionSwComponentType`. By this means, a hierarchical structures of software-components can be created. **Tags:** atp.recommendedPackage=SwComponentTypes			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *SwComponentType*			
Aggregated by	ARPackage.element			
Attribute	*Type*	*Mult.*	*Kind*	*Note*

▽

△

Class	**CompositionSwComponentType**			
component	SwComponent Prototype	*	aggr	The instantiated components that are part of this composition. The aggregation of SwComponentPrototype is subject to variability with the purpose to support the conditional existence of a SwComponentPrototype. Please be aware: if the conditional existence of SwComponentPrototypes is resolved post-build, the deselected SwComponentPrototypes are still contained in the ECUs build but the instances are inactive in that they are not scheduled by the RTE. The use case for having 0 components owned by the CompositionSwComponentType could be to deliver an empty CompositionSwComponentType to e.g. a supplier for filling the internal structure. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=component, component.shortName, component.variationPoint.shortLabel vh.latestBindingTime=postBuild
connector	SwConnector	*	aggr	SwConnectors have the principal ability to establish a connection among PortPrototypes. They can have many roles in the context of a CompositionSwComponentType. Details are refined by subclasses. The aggregation of SwConnectors is subject to variability with the purpose to support variant data flow. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=connector, connector.shortName, connector.variationPoint.shortLabel vh.latestBindingTime=postBuild
constantValue Mapping	ConstantSpecification MappingSet	*	ref	Reference to the ConstantSpecificationMapping to be applied for initValues of PPortComSpecs and RPortComSpec. **Tags:** atp.Splitkey=constantValueMapping
dataType Mapping	DataTypeMappingSet	*	ref	Reference to the DataTypeMappingSet to be applied for the used ApplicationDataTypes in PortInterfaces. Background: when developing subsystems it may happen that ApplicationDataTypes are used on the surface of CompositionSwComponentTypes. In this case it would be reasonable to be able to also provide the intended mapping to the ImplementationDataTypes. However, this mapping shall be informal and not technically binding for the implementors mainly because the RTE generator is not concerned about the CompositionSwComponentTypes. Rationale: if the mapping of ApplicationDataTypes on the delegated and inner PortPrototype matches then the mapping to ImplementationDataTypes is not impacting compatibility. **Tags:** atp.Splitkey=dataTypeMapping

▽

△

Class	**CompositionSwComponentType**			
instantiation RTEEventProps	InstantiationRTEEvent Props	*	aggr	This allows to define instantiation specific properties for RTE Events, in particular for instance specific scheduling. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=instantiationRTEEventProps, instantiation RTEEventProps.shortLabel, instantiationRTEEvent Props.variationPoint.shortLabel vh.latestBindingTime=codeGenerationTime This Attribute is only used by the AUTOSAR Classic Platform.
physical Dimension Mapping	PhysicalDimension MappingSet	0..1	ref	This reference identifies the `PhysicalDimensionMappingSet` that is applicable in the context of the enclosing `CompositionSwComponentType`. The `PhysicalDimensionMapping`s contained in the `PhysicalDimensionMappingSet` shall be taken into account for the assessment of the compatibility of `PhysicalDimension`s in the context of creation of a `PortInterfaceMapping` in the scope of the `CompositionSwComponentType`. **Tags:** atp.Splitkey=physicalDimensionMapping

Table E.9: CompositionSwComponentType

Class	**DocRevision**			
Note	This meta-class represents the ability to maintain information which relates to revision management of documents or objects.			
Base	*ARObject*			
Aggregated by	AdminData.docRevision			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
date	DateTime	0..1	attr	This specifies the date and time, when the object in question was released **Tags:** xml.sequenceOffset=80
issuedBy	String	0..1	attr	This is the name of an individual or an organization who issued the current revision of the document or document fragment. **Tags:** xml.sequenceOffset=60
modification	Modification	*	aggr	This property represents one particular modification in comparison to its predecessor. **Tags:** xml.roleElement=true xml.roleWrapperElement=true xml.sequenceOffset=100 xml.typeElement=false xml.typeWrapperElement=false
revisionLabel	RevisionLabelString	0..1	attr	This attribute represents the version number of the object. **Tags:** xml.sequenceOffset=20
revisionLabelP1	RevisionLabelString	0..1	attr	This attribute represents the version number of the first predecessor of the object. **Tags:** xml.sequenceOffset=30

▽

△

Class	**DocRevision**			
revisionLabelP2	RevisionLabelString	0..1	attr	This attribute represents the version number of the second predecessor of the object. This attribute is used if the object is the result of a merge process in which two branches are merged in to one new revision. **Tags:** xml.sequenceOffset=40
state	NameToken	0..1	attr	The attribute state represents the current state of the current file according to the configuration management plan. It is a NameToken until possible states are standardized. **Tags:** xml.sequenceOffset=50

Table E.10: DocRevision

Class	**Documentation**			
Note	This meta-class represents the ability to handle a so called standalone documentation. Standalone means, that such a documentation is not embedded in another AUTOSARElement or identifiable object. The standalone documentation is an entity of its own which denotes its context by reference to other objects and instances. **Tags:** atp.recommendedPackage=Documentations			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
context	DocumentationContext	*	aggr	This is the context of the particular documentation. **Tags:** atp.Splitkey=context, context.shortName
documentation Content	PredefinedChapter	0..1	aggr	This is the content of the documentation related to the specified contexts. **Tags:** atp.Splitkey=documentationContent xml.sequenceOffset=200

Table E.11: Documentation

Class	**EcuInstance**			
Note	ECUInstances are used to define the ECUs used in the topology. The type of the ECU is defined by a reference to an ECU specified with the ECU resource description. **Tags:** atp.recommendedPackage=EcuInstances			
Base	*ARObject*, *CollectableElement*, *FibexElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	EcuInstance			
associatedComIPduGroup	ISignalIPduGroup	*	ref	With this reference it is possible to identify which ISignal IPduGroups are applicable for which Communication Connector/ ECU. Only top level ISignalIPduGroups shall be referenced by an EcuInstance. If an ISignalIPduGroup contains other ISignalIPduGroups than these contained ISignalIPdu Groups shall not be referenced by the EcuInstance. Contained ISignalIPduGroups are associated to an Ecu Instance via the top level ISignalIPduGroup. **Tags:** atp.Splitkey=associatedComIPduGroup This Attribute is only used by the AUTOSAR Classic Platform.
associated ConsumedProvided ServiceInstance Group	ConsumedProvided ServiceInstanceGroup	*	ref	With this reference it is possible to identify which ConsumedProvidedServiceInstanceGroups are applicable for which ECUInstance. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=associatedConsumedProvidedService InstanceGroup, associatedConsumedProvidedService InstanceGroup.consumedProvidedServiceInstanceGroup, associatedConsumedProvidedServiceInstance Group.variationPoint.shortLabel vh.latestBindingTime=postBuild
associatedPdurIPduGroup	PdurIPduGroup	*	ref	With this reference it is possible to identify which PduR IPdu Groups are applicable for which Communication Connector/ ECU. **Tags:** atp.Splitkey=associatedPdurIPduGroup
channel Synchronous Wakeup	Boolean	0..1	attr	If this parameter is available and set to true, then all available channels will be woken up as soon as at least one channel wakeup occurs. If PNCs are configured, then all PNCs will be requested upon a channel wakeup.
clientIdRange	ClientIdRange	0..1	aggr	Restriction of the Client Identifier for this Ecu to an allowed range of numerical values. The Client Identifier of the transaction handle is generated by the client RTE for inter-Ecu Client/Server communication. **Tags:** atp.Splitkey=clientIdRange
com Configuration GwTimeBase	TimeValue	0..1	attr	The period between successive calls to Com_Main FunctionRouteSignals of the AUTOSAR COM module in seconds. This Attribute is only used by the AUTOSAR Classic Platform.
com ConfigurationRx TimeBase	TimeValue	0..1	attr	The period between successive calls to Com_Main FunctionRx of the AUTOSAR COM module in seconds. This Attribute is only used by the AUTOSAR Classic Platform.
com ConfigurationTx TimeBase	TimeValue	0..1	attr	The period between successive calls to Com_Main FunctionTx of the AUTOSAR COM module in seconds. This Attribute is only used by the AUTOSAR Classic Platform.
comEnable MDTForCyclic Transmission	Boolean	0..1	attr	Enables for the Com module of this EcuInstance the minimum delay time monitoring for cyclic and repeated transmissions (TransmissionModeTiming has cyclic Timing assigned or eventControlledTiming with numberOf Repetitions > 0). This Attribute is only used by the AUTOSAR Classic Platform.

▽

△

Class	EcuInstance			
commController	Communication Controller	*	aggr	CommunicationControllers of the ECU. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=commController, commController.shortName, commController.variationPoint.shortLabel vh.latestBindingTime=postBuild
connector	Communication Connector	*	aggr	All channels controlled by a single controller. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=connector, connector.shortName, connector.variationPoint.shortLabel vh.latestBindingTime=postBuild
dltConfig	DltConfig	0..1	aggr	Describes the Dlt configuration on this EcuInstance. **Tags:** atp.Splitkey=dltConfig This Attribute is only used by the AUTOSAR Classic Platform.
doIpConfig	DoIpConfig	0..1	aggr	DoIp configuration on this EcuInstance. **Tags:** atp.Splitkey=doIpConfig atp.Status=draft This Attribute is only used by the AUTOSAR Classic Platform.
ecuTaskProxy	OsTaskProxy	*	ref	Reference to OsTaskProxies assigned to the Ecu Instance. **Tags:** atp.Splitkey=ecuTaskProxy This Attribute is only used by the AUTOSAR Classic Platform.
ethSwitchPort Group Derivation	Boolean	0..1	attr	Defines whether the derivation of SwitchPortGroups based on VLAN and/or CouplingPort.pncMapping shall be performed for this EcuInstance. If not defined the derivation shall not be done. This Attribute is only used by the AUTOSAR Classic Platform.
firewallRule	StateDependentFirewall	*	ref	Firewall rules defined in the context of an EcuInstance. **Tags:** atp.Splitkey=firewallRule
j1939Node	J1939Node	*	aggr	Optional collection of J1939Nodes defined on this Ecu Instance. **Tags:** atp.Splitkey=j1939Node, j1939Node.shortName This Attribute is only used by the AUTOSAR Classic Platform.
partition	EcuPartition	*	aggr	Optional definition of Partitions within an Ecu. **Tags:** atp.Splitkey=partition, partition.shortName This Attribute is only used by the AUTOSAR Classic Platform.
pncNmRequest	Boolean	0..1	attr	Defines if this EcuInstance shall request Nm on all its PhysicalChannels which have Nm variant set to FULL each time a PNC is requested.
pncPrepare SleepTimer	TimeValue	0..1	attr	Time in seconds the PNC state machine shall wait in PNC_PREPARE_SLEEP.
pnc Synchronous Wakeup	Boolean	0..1	attr	If this parameter is available and set to true then all available PNCs will be woken up as soon as a channel wakeup occurs. This is ensured by adding all PNCs to all channel wakeup sources during upstream mapping.

▽

△

Class	**EcuInstance**			
pnResetTime	TimeValue	0..1	attr	Specifies the runtime of the reset timer in seconds. This reset time is valid for the reset of PN requests in the EIRA and in the ERA.
sleepMode Supported	Boolean	0..1	attr	Specifies whether the ECU instance may be put to a "low power mode" • true: sleep mode is supported • false: sleep mode is not supported Note: This flag may only be set to "true" if the feature is supported by both hardware and basic software. This Attribute is only used by the AUTOSAR Classic Platform.
tcpIpIcmpProps	EthTcpIpIcmpProps	0..1	ref	EcuInstance specific ICMP (Internet Control Message Protocol) attributes **Tags:** atp.Splitkey=tcpIpIcmpProps This Attribute is only used by the AUTOSAR Classic Platform.
tcpIpProps	EthTcpIpProps	0..1	ref	EcuInstance specific TcpIp Stack attributes. **Tags:** atp.Splitkey=tcpIpProps This Attribute is only used by the AUTOSAR Classic Platform.
wakeUpOver BusSupported	Boolean	0..1	attr	Driver support for wakeup over Bus. This Attribute is only used by the AUTOSAR Classic Platform.

Table E.12: EcuInstance

Class	***EcucAbstractConfigurationClass*** (abstract)			
Note	Specifies the ValueConfigurationClass of a parameter/reference or the MultiplicityConfigurationClass of a parameter/reference or a container for each ConfigurationVariant of the EcucModuleDef. **Tags:** atp.Status=obsolete This Class is only used by the AUTOSAR Classic Platform.			
Base	*ARObject*			
Subclasses	EcucMultiplicityConfigurationClass, EcucValueConfigurationClass			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
configClass	EcucConfigurationClass Enum	0..1	attr	Specifies the ConfigurationClass for the given ConfigurationVariant. **Tags:** atp.Status=obsolete
configVariant	EcucConfiguration VariantEnum	0..1	attr	Specifies the ConfigurationVariant the ConfigurationClass is specified for. **Tags:** atp.Status=obsolete

Table E.13: EcucAbstractConfigurationClass

Enumeration	**EcucConfigurationClassEnum**
Note	Possible configuration classes for the AUTOSAR configuration parameters. **Tags:** atp.Status=obsolete This Enumeration is only used by the AUTOSAR Classic Platform.
Aggregated by	*EcucAbstractConfigurationClass*.configClass

▽

△

Enumeration	**EcucConfigurationClassEnum**
Literal	***Description***
Link	Link Time: parts of configuration are delivered from another object code file **Tags:** atp.EnumerationLiteralIndex=0 atp.Status=obsolete
PostBuild	PostBuildTime: after compilation a configuration parameter can be changed. **Tags:** atp.EnumerationLiteralIndex=1 atp.Status=obsolete
PreCompile	PreCompile Time: after compilation a configuration parameter can not be changed any more. **Tags:** atp.EnumerationLiteralIndex=2 atp.Status=obsolete
Published Information	PublishedInformation is used to specify the fact that certain information is fixed even before the pre-compile stage. **Tags:** atp.EnumerationLiteralIndex=3 atp.Status=obsolete

Table E.14: EcucConfigurationClassEnum

Enumeration	**EcucConfigurationVariantEnum**
Note	Specifies the possible Configuration Variants used for AUTOSAR BSW Modules. **Tags:** atp.Status=obsolete This Enumeration is only used by the AUTOSAR Classic Platform.
Aggregated by	*EcucAbstractConfigurationClass*.configVariant, EcucModuleConfigurationValues.implementation ConfigVariant, EcucModuleDef.supportedConfigVariant
Literal	***Description***
Preconfigured Configuration	Preconfigured (i.e. fixed) configuration which cannot be changed. **Tags:** atp.EnumerationLiteralIndex=0 atp.Status=obsolete
Recommended Configuration	Recommended configuration for a module. **Tags:** atp.EnumerationLiteralIndex=1 atp.Status=obsolete
VariantLinkTime	Specifies that the BSW Module implementation may use PreCompileTime and LinkTime configuration parameters. **Tags:** atp.EnumerationLiteralIndex=2 atp.Status=obsolete
VariantPostBuild	Specifies that the BSW Module implementation may use PreCompileTime, LinkTime and PostBuild configuration parameters. **Tags:** atp.EnumerationLiteralIndex=3 atp.Status=obsolete
VariantPreCompile	Specifies that the BSW Module implementation uses only PreCompileTime configuration parameters. **Tags:** atp.EnumerationLiteralIndex=6 atp.Status=obsolete

Table E.15: EcucConfigurationVariantEnum

Class	EvaluatedVariantSet			
Note	This meta class represents the ability to express if a set of ARElements is able to support one or more particular variants. In other words, for a given set of evaluatedElements this meta class represents a table of evaluated variants, where each PredefinedVariant represents one column. In this column each descendant sw SystemconstantValue resp. postbuildVariantCriterionValue represents one entry. In a graphical representation each swSystemconstantValueSet / postBuildVariantCriterionValueSet could be used as an intermediate headline in the table column. If the approvalStatus is "APPROVED" it expresses that the collection of CollectableElements is known be valid for the given evaluatedVariants. Note that the EvaluatedVariantSet is a CollectableElement. This allows to establish a hierarchy of EvaluatedVariantSets. **Tags:** atp.recommendedPackage=EvaluatedVariantSets			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
approvalStatus	NameToken	0..1	attr	Defines the approval status of a predefined variant. Two values are predefined: "APPROVED" and "REJECTED": • Approved variants are known to work. • Rejected variants are known NOT to work. Further values can be approved on a per-company basis; within AUTOSAR only "APPROVED" and "REJECTED" should be recognized.
evaluated Element	CollectableElement	*	ref	This represents a particular element which is evaluated in context of the EvaluatedVariants. The approvalStatus applies to this element (and all of its descendants). In other words, the referenced elements are those that were considered when the predefined variant was evaluated. **Tags:** atp.Splitkey=evaluatedElement
evaluated Variant	PredefinedVariant	*	ref	This metaclass represents one particular variant which was evaluated. LowerMultiplicity is set to 0 to support a stepwise approach. **Tags:** atp.Splitkey=evaluatedVariant

Table E.16: EvaluatedVariantSet

Class	***Frame*** (abstract)			
Note	Data frame which is sent over a communication medium. This element describes the pure Layout of a frame sent on a channel.			
Base	*ARObject*, *CollectableElement*, *FibexElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Subclasses	*AbstractEthernetFrame*, CanFrame, FlexrayFrame, *LinFrame*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
frameLength	Integer	0..1	attr	The used length (in bytes) of the referencing frame. Should not be confused with a static byte length reserved for each frame by some platforms (e.g. FlexRay). The frameLength of zero bytes is allowed. Please consider also TPS_SYST_02255.



△

Class	Frame (abstract)			
pduToFrame Mapping	PduToFrameMapping	*	aggr	A frames layout as a sequence of Pdus. atpVariation: The content of a frame can be variable. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=pduToFrameMapping, pduToFrame Mapping.shortName, pduToFrameMapping.variation Point.shortLabel vh.latestBindingTime=postBuild

Table E.17: Frame

Class	FrameTriggering (abstract)			
Note	The FrameTriggering describes the instance of a frame sent on a channel and defines the manner of triggering (timing information) and identification of a frame on the channel, on which it is sent. For the same frame, if FrameTriggerings exist on more than one channel of the same cluster the fan-out/ in is handled by the Bus interface.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	CanFrameTriggering, EthernetFrameTriggering, FlexrayFrameTriggering, LinFrameTriggering			
Aggregated by	*PhysicalChannel*.frameTriggering			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
frame	Frame	0..1	ref	One frame can be triggered several times, e.g. on different channels. If a frame has no frame triggering, it won't be sent at all. A frame triggering has assigned exactly one frame, which it triggers. **Tags:** atp.Splitkey=frame
framePort	FramePort	*	ref	References to the FramePort on every ECU of the system which sends and/or receives the frame. References for both the sender and the receiver side shall be included when the system is completely defined. **Tags:** atp.Splitkey=framePort
pduTriggering	PduTriggering	*	ref	This reference provides the relationship to the Pdu Triggerings that are implemented by the FrameTriggering. The reference is optional since no PduTriggering can be defined for NmPdus and XCP Pdus. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=pduTriggering, pduTriggering.pduTriggering, pduTriggering.variationPoint.shortLabel vh.latestBindingTime=postBuild

Table E.18: FrameTriggering

Class	HwElement			
Note	This represents the ability to describe Hardware Elements on an instance level. The particular types of hardware are distinguished by the category. This category determines the applicable attributes. The possible categories and attributes are defined in HwCategory. **Tags:** atp.recommendedPackage=HwElements			
Base	*ARElement*, *ARObject*, *CollectableElement*, *HwDescriptionEntity*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**

▽

△

Class	HwElement			
hwElement Connection	HwElementConnector	*	aggr	This represents one particular connection between two hardware elements. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=hwElementConnection, hwElement Connection.variationPoint.shortLabel vh.latestBindingTime=systemDesignTime xml.sequenceOffset=110
hwPinGroup	HwPinGroup	*	aggr	This aggregation is used to describe the connection facilities of a hardware element. Note that hardware element has no pins but only pingroups. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=hwPinGroup, hwPinGroup.shortName, hw PinGroup.variationPoint.shortLabel vh.latestBindingTime=systemDesignTime xml.sequenceOffset=90
nestedElement	HwElement	*	ref	This association is used to establish hierarchies of hw elements. Note that one particular HwElement can be target of this association only once. I.e. multiple instantiation of the same HwElement is not supported (at any hierarchy level). **Stereotypes:** atpVariation **Tags:** atp.Splitkey=nestedElement, nestedElement.hwElement, nestedElement.variationPoint.shortLabel vh.latestBindingTime=systemDesignTime xml.sequenceOffset=70

Table E.19: HwElement

Class	*Identifiable* (abstract)
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*, AppOsTask ProxyToEcuTaskProxyMapping, ApplicationEndpoint, ApplicationError, ApplicationPartitionToEcuPartition Mapping, AppliedStandard, AsynchronousServerCallResultPoint, *AtpBlueprint*, *AtpClassifier*, *Atp Feature*, AutosarOperationArgumentInstance, AutosarVariableInstance, *BinaryManifestAddressable Object*, BinaryManifestItemDefinition, *BinaryManifestResource*, BinaryManifestResourceDefinition, Block State, BswInternalTriggeringPoint, BswModuleDependency, *BuildActionEntity*, BuildActionEnvironment, CanSecSecureChannel, CanTpAddress, CanTpChannel, CanTpNode, Chapter, ClientIdDefinition, Client ServerOperation, Code, *CollectableElement*, ComManagementMapping, *CommConnectorPort*, *CommunicationConnector*, *CommunicationController*, Compiler, ConsistencyNeeds, ConsumedEvent Group, *CouplingElementAbstractDetails*, CouplingPort, *CouplingPortAbstractShaper*, *CouplingPort StructuralElement*, *CpSoftwareClusterResource*, CpSoftwareClusterResourceToApplicationPartition Mapping, CpSoftwareClusterToApplicationPartitionMapping, CpSoftwareClusterToEcuInstanceMapping, CpSoftwareClusterToResourceMapping, *CryptoServiceMapping*, CyclicHandlingComDataToOsTask ProxyMapping, DataPrototypeGroup, DataPrototypeTransformationPropsIdent, DataTransformation, *Dds AbstractServiceInstanceElementCp*, DdsCpDomain, DdsCpPartition, DdsCpQosProfile, DdsCpTopic, DependencyOnArtifact, *DiagEventDebounceAlgorithm*, DiagnosticAuthTransmitCertificateEvaluation, DiagnosticConnectedIndicator, DiagnosticDataElement, DiagnosticDebounceAlgorithmProps, Diagnostic ExtendedDataRecordElement, DiagnosticFunctionInhibitSource, DiagnosticParameterElement, *DiagnosticRoutineSubfunction*, DltApplication, DltArgument, DltArgumentProps, DltLogChannel, Dlt Message, DoIpInterface, DoIpLogicAddress, DoIpRoutingActivation, ECUMapping, *EOCExecutableEntity RefAbstract*, EcuPartition, EcuPartitionToCoreMapping, EcucContainerValue, *EcucDefinitionElement*, ▽

▽

△

Class	***Identifiable*** (abstract)			
	△ EcucDestinationUriDef, EcucEnumerationLiteralDef, EcucQuery, EcucValidationCondition, Ethernet WakeupSleepOnDatalineConfig, EventHandler, ExclusiveArea, *ExecutableEntity*, *ExecutionTime*, FMAttributeDef, FMFeatureMapAssertion, FMFeatureMapCondition, FMFeatureMapElement, FMFeature Relation, FMFeatureRestriction, FMFeatureSelection, FlatInstanceDescriptor, FlexrayTpConnection Control, FlexrayTpNode, FlexrayTpPduPool, *FrameTriggering*, GlobalTimeGateway, *GlobalTimeMaster*, *GlobalTimeSlave*, *HeapUsage*, HwAttributeDef, HwAttributeLiteralDef, HwPin, HwPinGroup, *IEEE1722 TpAcfBus*, *IEEE1722TpAcfBusPart*, IPSecRule, IPv6ExtHeaderFilterList, ISignalToIPduMapping, ISignal Triggering, *IdentCaption*, ImpositionTime, InternalTriggeringPoint, J1939Node, J1939SharedAddress Cluster, J1939TpNode, Keyword, LifeCycleState, LinScheduleTable, LinTpNode, Linker, MacAddress VlanMembership, MacMulticastGroup, MacSecKayParticipant, McDataInstance, MemorySection, Mode Declaration, ModeDeclarationMapping, ModeSwitchPoint, ModeSwitchSenderComSpecProps, Network Endpoint, *NmCluster*, NmEcu, *NmNode*, NvBlockDescriptor, *PackageableElement*, ParameterAccess, PduActivationRoutingGroup, PduToFrameMapping, PduTriggering, PerInstanceMemory, *Physical Channel*, PortElementToCommunicationResourceMapping, PortGroup, *PortInterfaceMapping*, Queued ReceiverComSpecProps, ResourceConsumption, RootSwCompositionPrototype, RptComponent, Rpt Container, RptExecutableEntity, RptExecutableEntityEvent, RptExecutionContext, RptProfile, RptService Point, RteEventInCompositionSeparation, RteEventInCompositionToOsTaskProxyMapping, RteEventIn SystemSeparation, RteEventInSystemToOsTaskProxyMapping, RunnableEntityGroup, *SdgAttribute*, Sdg Class, SecOcJobRequirement, SecureCommunicationAuthenticationProps, SecureCommunication FreshnessProps, SecurityEventContextDataElement, SecurityEventContextProps, *ServerCallPoint*, ServerComSpecProps, *ServiceNeeds*, SignalServiceTranslationElementProps, SignalServiceTranslation EventProps, SignalServiceTranslationProps, SocketAddress, SomeipTpChannel, *StackUsage*, Static SocketConnection, StructuredReq, SwGenericAxisParamType, SwServiceArg, SwcServiceDependency, SwcToApplicationPartitionMapping, SwcToEcuMapping, SwcToImplMapping, SwitchAsynchronousTraffic ShaperGroupEntry, SwitchAtsInstanceEntry, SwitchFlowMeteringEntry, SwitchStreamFilterActionDest PortModification, SwitchStreamFilterEntry, SwitchStreamFilterRule, SwitchStreamGateEntry, Switch StreamIdentification, SystemMapping, SystemSignalGroupToCommunicationResourceMapping, System SignalToCommunicationResourceMapping, TDCpSoftwareClusterMapping, TDCpSoftwareCluster ResourceMapping, TcpOptionFilterList, *TimingClock*, TimingClockSyncAccuracy, TimingCondition, *TimingConstraint*, *TimingDescription*, TimingExtensionResource, TimingModeInstance, TlsCryptoCipher Suite, TlsCryptoCipherSuiteProps, Topic1, TpAddress, TraceableTable, TraceableText, *TracedFailure*, TransformationISignalPropsIdent, *TransformationProps*, TransformationTechnology, Trigger, Variable Access, VariationPointProxy, ViewMap, VlanConfig, WaitPoint			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
adminData	AdminData	0..1	aggr	This represents the administrative data for the identifiable object. **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:** atp.Splitkey=annotation xml.sequenceOffset=-25
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

▽

△

Class	***Identifiable*** (abstract)			
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:** atp.Splitkey=desc 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:** atp.Splitkey=introduction 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 E.20: Identifiable

Primitive	**Identifier**			
Note	An Identifier is a string with a number of constraints on its appearance, satisfying the requirements typical programming languages define for their Identifiers. This datatype represents a string, that can be used as a c-Identifier. It shall start with a letter, may consist of letters, digits and underscores. **Tags:** xml.xsd.customType=IDENTIFIER xml.xsd.maxLength=128 xml.xsd.pattern=[a-zA-Z][a-zA-Z0-9_]* xml.xsd.type=string			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Primitive	**Identifier**			
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
namePattern	String	0..1	attr	This attribute represents a pattern which shall be used to define the value of the identifier if the identifier in question is part of a blueprint. For more details refer to TPS_StandardizationTemplate. **Tags:** xml.attribute=true

Table E.21: Identifier

Class	**ImplementationDataType**			
Note	Describes a reusable data type on the implementation level. This will typically correspond to a typedef in C-code. **Tags:** atp.recommendedPackage=ImplementationDataTypes			
Base	*ARElement*, *ARObject*, *AbstractImplementationDataType*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *Autosar DataType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dynamicArray SizeProfile	String	0..1	attr	Specifies the profile which the array will follow in case this data type is a variable size array.
isStructWith Optional Element	Boolean	0..1	attr	This attribute is only valid if the attribute category is set to STRUCTURE. If set to true, this attribute indicates that the ImplementationDataType has been created with the intention to define at least one element of the structure as optional.
subElement (ordered)	ImplementationData TypeElement	*	aggr	Specifies an element of an array, struct, or union data type. The aggregation of `ImplementationDataTypeElement` is subject to variability with the purpose to support the conditional existence of elements inside a `ImplementationDataType` representing a structure. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
symbolProps	SymbolProps	0..1	aggr	This represents the SymbolProps for the Implementation DataType. **Tags:** atp.Splitkey=symbolProps, symbolProps.short Name
typeEmitter	NameToken	0..1	attr	This attribute is used to control which part of the AUTOSAR toolchain is supposed to trigger data type definitions.

Table E.22: ImplementationDataType

Enumeration	**IntervalTypeEnum**
Note	This enumerator specifies the type of an interval.
Aggregated by	Limit.intervalType, LimitValueVariationPoint.intervalType
Literal	***Description***
closed	The area is limited by the value given. The value itself is included. **Tags:** atp.EnumerationLiteralIndex=0
open	The area is limited by the value given. The value itself is not included. **Tags:** atp.EnumerationLiteralIndex=2

Table E.23: IntervalTypeEnum

Class	**LifeCycleState**			
Note	This meta class represents one particular state in the LifeCycle.			
Base	*ARObject*, *AtpBlueprint*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	LifeCycleStateDefinitionGroup.lcState			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table E.24: LifeCycleState

Primitive	**Limit**			
Note	This class represents the ability to express a numerical limit. Note that this is in fact a NumericalVariation Point but has the additional attribute intervalType. **Tags:** xml.xsd.customType=LIMIT-VALUE xml.xsd.pattern=(0[xX][0-9a-fA-F]+)|(0[0-7]+)|(0[bB][0-1]+)|(([+\-]?[1-9][0-9]+(\.[0-9]+)?|[+\-]?[0-9](\.[0-9]+)?)([eE]([+\-]?)[0-9]+)?)|\.0|INF|-INF|NaN xml.xsd.type=string			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
intervalType	IntervalTypeEnum	0..1	attr	This specifies the type of the interval. If the attribute is missing the interval shall be considered as "CLOSED". **Tags:** xml.attribute=true

Table E.25: Limit

Class	**MIFormula**			
Note	This meta-class represents the ability to express a formula in a documentation. The formula can be expressed by various means. If more than one representation is available, they need to be consistent. The rendering system can use the representation which is most appropriate.			
Base	*ARObject*, *DocumentViewSelectable*, *Paginateable*			
Aggregated by	DocumentationBlock.formula, EcucConditionSpecification.informalFormula, EcucDerivation Specification.informalFormula			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
formulaCaption	Caption	0..1	aggr	This element specifies the identification or heading of a formula. **Tags:** atp.Splitkey=formulaCaption, formulaCaption.shortName xml.sequenceOffset=20

▽

△

Class	**MIFormula**			
genericMath	MultiLanguagePlainText	0..1	aggr	this rpresents the semantic and mathematical descriptions which are processed by a math-processor. **Tags:** atp.Splitkey=genericMath xml.sequenceOffset=80
lGraphic	LGraphic	*	aggr	This represents a formula as an embedded figure. **Tags:** xml.roleWrapperElement=false xml.sequenceOffset=30
texMath	MultiLanguagePlainText	0..1	aggr	this is the TeX representation of TeX formula. A TeX formula can be processed by a TeX or a LaTeX processor. **Tags:** atp.Splitkey=texMath xml.sequenceOffset=60
verbatim	MultiLanguageVerbatim	0..1	aggr	this represents a formula using only text and white-space. It can be used to denote the formula in a kind of pseudo code or whatever appears approprate. **Tags:** atp.Splitkey=verbatim xml.sequenceOffset=50

Table E.26: MIFormula

Class	***MultilanguageReferrable*** (abstract)			
Note	Instances of this class can be referred to by their identifier (while adhering to namespace borders). They also may have a longName. But they are not considered to contribute substantially to the overall structure of an AUTOSAR description. In particular it does not contain other Referrables.			
Base	*ARObject*, *Referrable*			
Subclasses	Caption, DefItem, DocumentationContext, *Identifiable*, SdgCaption, *TraceReferrable*, *Traceable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
longName	MultilanguageLong Name	0..1	aggr	This specifies the long name of the object. Long name is targeted to human readers and acts like a headline. **Tags:** atp.Splitkey=longName

Table E.27: MultilanguageReferrable

Class	**NPdu**			
Note	This is a Pdu of the Transport Layer. The main purpose of the TP Layer is to segment and reassemble IPdus. **Tags:** atp.recommendedPackage=Pdus			
Base	*ARElement*, *ARObject*, *CollectableElement*, *FibexElement*, *IPdu*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Pdu*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table E.28: NPdu

Class	**NmPdu**			
Note	Network Management Pdu **Tags:** atp.recommendedPackage=Pdus			
Base	*ARElement*, *ARObject*, *CollectableElement*, *FibexElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Pdu*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
iSignalToIPdu Mapping	ISignalToIPduMapping	*	aggr	This optional aggregation is used to describe NmUser Data that is transmitted in the NmPdu. The counting of the startPosition starts at the beginning of the NmPdu regardless whether Cbv or Nid are used. **Tags:** atp.Splitkey=iSignalToIPduMapping, iSignalToIPdu Mapping.shortName
nmData Information	Boolean	0..1	attr	Defines if the Pdu contains NM Data. If the NmPdu does not aggregate any ISignalToIPduMappings it still may contain UserData that is set via Nm_SetUserData(). If the ISignalToIPduMapping exists then the nmDataInformation attribute shall be ignored.
nmVote Information	Boolean	0..1	attr	Defines if the Pdu contains NM Vote information.
unusedBit Pattern	Integer	0..1	attr	AUTOSAR COM is filling not used areas of an Pdu with this bit-pattern. This attribute can only be used if the nm DataInformation attribute is set to true.

Table E.29: NmPdu

Class	«atpMixedString» **NumericalValueVariationPoint**			
Note	This class represents an attribute value variation point for Numerical attributes. Note that this class might be used in the extended meta-model only.			
Base	*ARObject*, *AbstractNumericalVariationPoint*, *AttributeValueVariationPoint*, *FormulaExpression*, *SwSystemconstDependentFormula*			
Aggregated by	VariationPointProxy.valueAccess			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table E.30: NumericalValueVariationPoint

Class	***PackageableElement*** (abstract)			
Note	This meta-class specifies the ability to be a member of an AUTOSAR package.			
Base	*ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	*ARElement*, EnumerationMappingTable, *FibexElement*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table E.31: PackageableElement

Class	***Pdu*** (abstract)
Note	Collection of all Pdus that can be routed through a bus interface.
Base	*ARElement*, *ARObject*, *CollectableElement*, *FibexElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*

▽

△

Class	***Pdu*** (abstract)			
Subclasses	GeneralPurposePdu, *IPdu*, NmPdu, UserDefinedPdu			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
hasDynamic Length	Boolean	0..1	attr	This attribute defines whether the Pdu has dynamic length (true) or not (false). Please note that the usage of this attribute is restricted by [constr_3448].
length	UnlimitedInteger	0..1	attr	Pdu length in bytes. In case of dynamic length IPdus (containing a dynamical length signal), this value indicates the maximum data length. It should be noted that in former AUTOSAR releases (Rel 2.1, Rel 3.0, Rel 3.1, Rel 4.0 Rev. 1) this parameter was defined in bits. The Pdu length of zero bytes is allowed.

Table E.32: Pdu

Class	**PduTriggering**			
Note	The PduTriggering describes on which channel the IPdu is transmitted. The Pdu routing by the PduR is only allowed for subclasses of IPdu. Depending on its relation to entities such channels and clusters it can be unambiguously deduced whether a fan-out is handled by the Pdu router or the Bus Interface. If the fan-out is specified between different clusters it shall be handled by the Pdu Router. If the fan-out is specified between different channels of the same cluster it shall be handled by the Bus Interface.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	*PhysicalChannel*.pduTriggering			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
iPdu	Pdu	0..1	ref	Reference to the Pdu for which the PduTriggering is defined. One I-Pdu can be triggered on different channels (PduR fan-out). The Pdu routing by the PduR is only allowed for subclasses of IPdu. Nevertheless is the reference to the Pdu element necessary since the PduTriggering element is also used to specify the sending and receiving connections to Ecu Ports. **Tags:** atp.Splitkey=iPdu
iPduPort	IPduPort	*	ref	References to the IPduPort on every ECU of the system which sends and/or receives the I-PDU. References for both the sender and the receiver side shall be included when the system is completely defined. **Tags:** atp.Splitkey=iPduPort
iSignal Triggering	ISignalTriggering	*	ref	This reference provides the relationship to the ISignal Triggerings that are implemented by the PduTriggering. The reference is optional since no ISignalTriggering can be defined for DCM and Multiplexed Pdus. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=iSignalTriggering, iSignalTriggering.iSignal Triggering, iSignalTriggering.variationPoint.shortLabel vh.latestBindingTime=postBuild

▽

△

Class	**PduTriggering**			
secOcCrypto Mapping	SecOcCryptoService Mapping	0..1	ref	This reference identifies the crypto profile applicable to the usage (send, receive) of the also referenced Secured IPdu. Obviously, this reference is only applicable if the Pdutriggering also references a SecuredIPdu in the role i Pdu. **Tags:** atp.Splitkey=secOcCryptoMapping atp.Status=obsolete
secOcCrypto ServiceMapping	SecOcCryptoService Mapping	0..1	ref	This reference identifies the crypto profile applicable to the usage (send, receive) of the also referenced Secured IPdu. Obviously, this reference is only applicable if the Pdutriggering also references a SecuredIPdu in the role i Pdu. **Stereotypes:** atpVariation **Tags:** atp.Splitkey=secOcCryptoServiceMapping, secOcCrypto ServiceMapping.secOcCryptoServiceMapping, secOc CryptoServiceMapping.variationPoint.shortLabel vh.latestBindingTime=postBuild
triggerIPduSend Condition	TriggerIPduSend Condition	*	aggr	Defines the trigger for the Com_TriggerIPDUSend API call. Only if all defined TriggerIPduSendConditions evaluate to true (AND associated) the Com_Trigger IPDUSend API shall be called. **Tags:** atp.Splitkey=triggerIPduSendCondition

Table E.33: PduTriggering

Class	***PortPrototype*** (abstract)			
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*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	*AbstractProvidedPortPrototype*, *AbstractRequiredPortPrototype*			
Aggregated by	*AtpClassifier*.atpFeature, *SwComponentType*.port			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
clientServer Annotation	ClientServerAnnotation	*	aggr	Annotation of this `PortPrototype` with respect to client/ server communication. **Tags:** atp.Splitkey=clientServerAnnotation
delegatedPort Annotation	DelegatedPort Annotation	0..1	aggr	Annotations on this delegated port. **Tags:** atp.Splitkey=delegatedPortAnnotation
ioHwAbstraction Server Annotation	IoHwAbstractionServer Annotation	*	aggr	Annotations on this IO Hardware Abstraction port. **Tags:** atp.Splitkey=ioHwAbstractionServerAnnotation
modePort Annotation	ModePortAnnotation	*	aggr	Annotations on this mode port. **Tags:** atp.Splitkey=modePortAnnotation
nvDataPort Annotation	NvDataPortAnnotation	*	aggr	Annotations on this non voilatile data port. **Tags:** atp.Splitkey=nvDataPortAnnotation
parameterPort Annotation	ParameterPort Annotation	*	aggr	Annotations on this parameter port. **Tags:** atp.Splitkey=parameterPortAnnotation

▽

△

Class	***PortPrototype*** (abstract)			
senderReceiver Annotation	SenderReceiver Annotation	*	aggr	Collection of annotations of this ports sender/receiver communication. **Tags:** atp.Splitkey=senderReceiverAnnotation
triggerPort Annotation	TriggerPortAnnotation	*	aggr	Annotations on this trigger port. **Tags:** atp.Splitkey=triggerPortAnnotation

Table E.34: PortPrototype

Class	**PostBuildVariantCriterion**			
Note	This class specifies one particular PostBuildVariantSelector. **Tags:** atp.recommendedPackage=PostBuildVariantCriterions			
Base	*ARElement*, *ARObject*, *AtpDefinition*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
compuMethod	CompuMethod	0..1	ref	The compuMethod specifies the possible values for the variant criterion serving as an enumerator. **Tags:** atp.Splitkey=compuMethod

Table E.35: PostBuildVariantCriterion

Class	**PostBuildVariantCriterionValue**			
Note	This class specifies the value which shall be assigned to a particular variant criterion in order to bind the variation point. If multiple criterion/value pairs are specified, they all shall match to bind the variation point.			
Base	*ARObject*			
Aggregated by	PostBuildVariantCriterionValueSet.postBuildVariantCriterionValue			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
annotation	Annotation	*	aggr	This provides the ability to add information why the value is set like it is. **Tags:** atp.Splitkey=annotation xml.sequenceOffset=30
value	Integer	0..1	attr	This is the particular value of the post-build variant criterion. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=20
variantCriterion	PostBuildVariant Criterion	0..1	ref	This association selects the variant criterion whose value is specified. **Tags:** atp.Splitkey=variantCriterion xml.sequenceOffset=10

Table E.36: PostBuildVariantCriterionValue

Class	PredefinedVariant			
Note	This specifies one predefined variant. It is characterized by the union of all system constant values and post-build variant criterion values aggregated within all referenced system constant value sets and post build variant criterion value sets plus the value sets of the included variants. **Tags:** atp.recommendedPackage=PredefinedVariants			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
includedVariant	PredefinedVariant	*	ref	The associated variants are considered part of this PredefinedVariant. This means the settings of the included variants are included in the settings of the referencing PredefinedVariant. Nevertheless the included variants might be included in several predefined variants. **Tags:** atp.Splitkey=includedVariant
postBuildVariant CriterionValue Set	PostBuildVariant CriterionValueSet	*	ref	This is the postBuildVariantCriterionValueSet contributing to the predefinded variant. **Tags:** atp.Splitkey=postBuildVariantCriterionValueSet
sw Systemconstant ValueSet	SwSystemconstant ValueSet	*	ref	This ist the set of Systemconstant Values contributing to the predefined variant. **Tags:** atp.Splitkey=swSystemconstantValueSet

Table E.37: PredefinedVariant

Class	***Referrable*** (abstract)			
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*, LinSlaveConfig Ident, ModeTransition, *MultilanguageReferrable*, PncMappingIdent, *SingleLanguageReferrable*, SoCon IPduIdentifier, 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:** atp.Splitkey=shortNameFragment xml.sequenceOffset=-90

Table E.38: Referrable

Class	***ServiceNeeds*** (abstract)
Note	This expresses the abstract needs that a Software Component or Basic Software Module has on the configuration of an AUTOSAR Service to which it will be connected. "Abstract needs" means that the model abstracts from the Configuration Parameters of the underlying Basic Software.
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*

▽

△

Class	**ServiceNeeds** (abstract)			
Subclasses	BswMgrNeeds, ChargeManagerNeeds, ComMgrUserNeeds, CryptoKeyManagementNeeds, CryptoServiceJobNeeds, CryptoServiceNeeds, *DiagnosticCapabilityElement*, DltUserNeeds, *DoIpServiceNeeds*, EcuStateMgrUserNeeds, ErrorTracerNeeds, FunctionInhibitionAvailabilityNeeds, FunctionInhibitionNeeds, GeneralPurposeTimerServiceNeeds, GlobalSupervisionNeeds, IdsMgrCustomTimestampNeeds, IdsMgrNeeds, IndicatorStatusNeeds, J1939DcmDm19Support, J1939RmIncomingRequestServiceNeeds, J1939RmOutgoingRequestServiceNeeds, NvBlockNeeds, SecureOnBoardCommunicationNeeds, SupervisedEntityCheckpointNeeds, SupervisedEntityNeeds, SyncTimeBaseMgrUserNeeds, VendorSpecificServiceNeeds			
Aggregated by	BswServiceDependency.serviceNeeds, SwcServiceDependency.serviceNeeds			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table E.39: ServiceNeeds

Class	**ServiceSwComponentType**			
Note	ServiceSwComponentType is used for configuring services for a given ECU. Instances of this class are only to be created in ECU Configuration phase for the specific purpose of the service configuration. **Tags:** atp.recommendedPackage=SwComponentTypes			
Base	*ARElement*, *ARObject*, *AtomicSwComponentType*, *AtpBlueprint*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *SwComponentType*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table E.40: ServiceSwComponentType

Class	**SwComponentPrototype**			
Note	Role of a software component within a composition.			
Base	*ARObject*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, CompositionSwComponentType.component			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
type	SwComponentType	0..1	tref	Type of the instance. **Stereotypes:** isOfType **Tags:** atp.Splitkey=type

Table E.41: SwComponentPrototype

Class	**SwConnector** (abstract)			
Note	The base class for connectors between ports. Connectors have to be identifiable to allow references from the system constraint template.			
Base	*ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	AssemblySwConnector, DelegationSwConnector, PassThroughSwConnector			
Aggregated by	*AtpClassifier*.atpFeature, CompositionSwComponentType.connector			
Attribute	**Type**	**Mult.**	**Kind**	**Note**

▽

△

Class	***SwConnector*** (abstract)			
mapping	PortInterfaceMapping	0..1	ref	Reference to a `PortInterfaceMapping` specifying the mapping of unequal named `PortInterface` elements of the two different `PortInterface`s typing the two `PortPrototype`s which are referenced by the `SwConnector`. **Tags:** atp.Splitkey=mapping

Table E.42: SwConnector

Class	**SwSystemconst**			
Note	This element defines a system constant which serves an input to select a particular variation point. In particular a system constant serves as an operand of the binding function (swSyscond) in a Variation point. Note that the binding process can only happen if a value was assigned to to the referenced system constants. **Tags:** atp.recommendedPackage=SwSystemconsts			
Base	*ARElement*, *ARObject*, *AtpDefinition*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
swDataDef Props	SwDataDefProps	0..1	aggr	This denotes the data definition properties of the system constant. This supports to express the limits and optionally a conversion within the internal to physical values by a compu method. **Tags:** atp.Splitkey=swDataDefProps xml.sequenceOffset=40

Table E.43: SwSystemconst

Class	**SwSystemconstantValueSet**			
Note	This meta-class represents the ability to specify a set of system constant values. **Tags:** atp.recommendedPackage=SwSystemconstantValueSets			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
sw Systemconstant Value	SwSystemconstValue	*	aggr	This is one particular value of a system constant. **Tags:** atp.Splitkey=swSystemconstantValue

Table E.44: SwSystemconstantValueSet

Class	UnitGroup			
Note	This meta-class represents the ability to specify a logical grouping of units.The category denotes the unit system that the referenced units are associated to. In this way, e.g. country-specific unit systems (CATEGORY="COUNTRY") can be defined as well as specific unit systems for certain application domains. In the same way a group of equivalent units, can be defined which are used in different countries, by setting CATEGORY="EQUIV_UNITS". KmPerHour and MilesPerHour could such be combined to one group named "vehicle_speed". The unit MeterPerSec would not belong to this group because it is normally not used for vehicle speed. But all of the mentioned units could be combined to one group named "speed". Note that the UnitGroup does not ensure the physical compliance of the units. This is maintained by the physical dimension. **Tags:** atp.recommendedPackage=UnitGroups			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
unit	Unit	*	ref	This represents one particular unit in the UnitGroup. **Tags:** atp.Splitkey=unit xml.sequenceOffset=20

Table E.45: UnitGroup

Class	UserDefinedPdu			
Note	UserDefinedPdu allows to describe PDU-based communication over Complex Drivers. If a new BSW module is added above the BusIf (e.g. a new Nm module) then this Pdu element shall be used to describe the communication. **Tags:** atp.recommendedPackage=Pdus			
Base	*ARElement*, *ARObject*, *CollectableElement*, *FibexElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Pdu*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
cddType	String	0..1	attr	This attribute defines the CDD that transmits or receives the UserDefinedIPdu. If several CDDs are defined this attribute is used to distinguish between them.

Table E.46: UserDefinedPdu

Class	VariationPoint			
Note	This meta-class represents the ability to express a "structural variation point". The container of the variation point is part of the selected variant if swSyscond evaluates to true and each postBuildVariant Criterion is fulfilled.			
Base	*ARObject*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
blueprint Condition	DocumentationBlock	0..1	aggr	This represents a description that documents how the variation point shall be resolved when deriving objects from the blueprint. Note that variationPoints are not allowed within a blueprintCondition. **Tags:** xml.sequenceOffset=28

▽

△

Class	**VariationPoint**			
desc	MultiLanguageOverview Paragraph	0..1	aggr	This allows to describe shortly the purpose of the variation point. **Tags:** xml.sequenceOffset=20
formalBlueprint Generator	BlueprintGenerator	0..1	aggr	This represents a description that documents how the variation point shall be resolved when deriving objects from the blueprint by using ARMQL. Note that variationPoints are not allowed within a formal BlueprintGenerator. **Tags:** atp.Status=draft xml.sequenceOffset=30
postBuildVariant Condition	PostBuildVariant Condition	*	aggr	This is the set of post build variant conditions which all shall be fulfilled in order to (postbuild) bind the variation point. **Tags:** xml.sequenceOffset=40
sdg	Sdg	0..1	aggr	An optional special data group is attached to every variation point. These data can be used by external software systems to attach application specific data. For example, a variant management system might add an identifier, an URL or a specific classifier. **Tags:** xml.sequenceOffset=50
shortLabel	Identifier	0..1	attr	This provides a name to the particular variation point to support the RTE generator. It is necessary for supporting splitable aggregations and if binding time is later than codeGenerationTime, as well as some RTE conditions. It needs to be unique with in the enclosing Identifiables with the same ShortName. **Stereotypes:** atpIdentityContributor **Tags:** xml.sequenceOffset=10
swSyscond	ConditionByFormula	0..1	aggr	This condition acts as Binding Function for the Variation Point. Note that the multiplicity is 0..1 in order to support pure postBuild variants. **Tags:** xml.sequenceOffset=30

Table E.47: VariationPoint

Primitive	**VerbatimString**			
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 E.48: VerbatimString

F Non-Splitable Elements in the Scope of this Document

This chapter contains a table of all model elements which are **not splitable** in the scope of this document.

Note for R26-11:

The life cycle of the stereotype ≪atpSplitable≫ will end with the AUTOSAR Release R26-11. Any remaining references to the stereotype ≪atpSplitable≫ in this document will be removed in release R27-05.

Name of non splitable element
AnyInstanceRef.base
AnyInstanceRef.contextElement
AnyInstanceRef.target
AtpClassifier.atpFeature
AtpInstanceRef.atpBase
AtpInstanceRef.atpContextElement
AtpInstanceRef.atpTarget
AtpPrototype.atpType
EcucConditionFormula.ecucQuery
EcucConditionFormula.ecucQueryString
EcucEnumerationParamDef.defaultValue
EcucParameterDerivationFormula.ecucQuery
EcucParameterDerivationFormula.ecucQueryString
EcucQueryExpression.configElementDefGlobal
EcucQueryExpression.configElementDefLocal
FormulaExpression.atpReference
FormulaExpression.atpStringReference
SdgClass.attribute
SwCalprmAxis.swCalprmAxisTypeProps
SwCalprmRefProxy.arParameter
SwCalprmRefProxy.mcDataInstance
SwDataDependencyArgs.swCalprmRef
SwDataDependencyArgs.swVariable
SwRecordLayoutGroup.swRecordLayoutGroupContentType
SwRecordLayoutGroupContent.swRecordLayout
SwRecordLayoutGroupContent.swRecordLayoutGroup
SwRecordLayoutGroupContent.swRecordLayoutV
SwRecordLayoutV.baseType
SwRecordLayoutV.desc
SwRecordLayoutV.swGenericAxisParamType

Table F.1: Usage of non splitable elements

G Variation Points in the Scope of this Document

This chapter contains a table of all model elements stereotyped ≪atpVariation≫ in the scope of this document.

Each entry in the table consists of the identification of the model element itself and the applicable value of the tagged value vh.latestBindingTime.

For more information about the concept of variation points and how model elements that contain variation points shall be treated please refer to [4].

Variation Point	*Latest Binding Time*
ARPackage.arPackage	blueprintDerivationTime
ARPackage.element	systemDesignTime
EcucAbstractStringParamDef	codeGenerationTime
EcucBooleanParamDef.defaultValue	codeGenerationTime
EcucContainerValue.parameterValue	postBuild
EcucContainerValue.referenceValue	postBuild
EcucContainerValue.subContainer	postBuild
EcucDefinitionElement.lowerMultiplicity	codeGenerationTime
EcucDefinitionElement.upperMultiplicity	codeGenerationTime
EcucDefinitionElement.upperMultiplicityInfinite	codeGenerationTime
EcucFloatParamDef.defaultValue	codeGenerationTime
EcucFloatParamDef.max	codeGenerationTime
EcucFloatParamDef.min	codeGenerationTime
EcucFunctionNameDef	codeGenerationTime
EcucIntegerParamDef.defaultValue	codeGenerationTime
EcucIntegerParamDef.max	codeGenerationTime
EcucIntegerParamDef.min	codeGenerationTime
EcucLinkerSymbolDef	codeGenerationTime
EcucModuleConfigurationValues.container	postBuild
EcucMultilineStringParamDef	codeGenerationTime
EcucNumericalParamValue.value	preCompileTime
EcucStringParamDef	codeGenerationTime
EcucValueCollection.ecucValue	preCompileTime
SwDataDefProps	codeGenerationTime
SwDataDefProps.swValueBlockSize	preCompileTime
SwDataDefProps.swValueBlockSizeMult	preCompileTime
SwTextProps.swMaxTextSize	preCompileTime
ValueList.vf	preCompileTime

Table G.1: Usage of variation points