

Document Title	Software Component Template
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	062
Document Classification	Standard

Document Status	Final
Part of AUTOSAR Release	4.2.2

Document Change History		
Release	Changed by	Description
4.2.2	AUTOSAR Release Management	Minor corrections / clarifications / editorial changes; For details please refer to the ChangeDocumentation
4.2.1	AUTOSAR Release Management	- Efficient NV data handling - Introduction of data transformation - Support for variable-size Arrays of arbitrary data types - Support for ASIL/QM development - Minor corrections / clarifications / editorial changes; For details please refer to the BWCSStatement
4.1.3	AUTOSAR Release Management	Various fixes and clarifications
4.1.2	AUTOSAR Release Management	Various fixes and clarifications

4.1.1	AUTOSAR Administration	• Introduction of PRPortPrototype • Definition of implicit communication behavior • Support for the formal analysis of resource locking • Introduction of refined scheduling of RunnableEntitys • Get information about activating RTEEvent • Connection of Mode Managers and Mode Users with different number of ModeDeclarations • Support activation of RunnableEntitys on remote ECUs • Support for ModeTransition • Support for the definition of the network representation of composite data types • ServiceNeeds for diagnostics over IP • Various fixes and clarifications
4.0.3	AUTOSAR Administration	• Added CompuMethod categories SCALE_LINEAR_AND_TEXTTABLE and SCALE_RATIONAL_AND_TEXTTABLE (table 5.73) • Clarification concerning the usage of invalid values • Revised support for data filters • Support for partial networking • Support for the specification of local connections between software-components • Improved description of service needs • Change history of constraints and specification items • Miscellaneous improvements and clarifications • “Support for Standardization” moved to Standardization Template [1]
4.0.2	AUTOSAR Administration	• Remove restriction on data type of inter-runnable variables • Rework end-to-end communication protection • Add more constraints on the usage of the meta-model • Various fixes and clarifications

4.0.1	AUTOSAR Administration	• New requirements tracing table • Support for fixed data exchange • Implementation of meta-model cleanup • Fundamental revision of the data type concept • Support for variant handling • Support for end-to-end communication protection • Support for documentation • Support for stopping and restarting of software-components • Support for triggered events • Support for explicit mapping of interface elements • Revised concept of mode management • Support for integrity and scaling at ports • Support for standardization within AUTOSAR
3.1.1	AUTOSAR Administration	• Improved support for on-board diagnostics • Small layout adaptations made
3.0.1	AUTOSAR Administration	• Improved support for measurement and calibration • Improved semantics of delegation ports • Introduction of abstract memory classes • Document meta information extended • Small layout adaptations made
2.1.15	AUTOSAR Administration	• Harmonization of the document with other specifications (e.g. RTE) • Introduction of a new concept to support calibration and measurement - harmonized with RTE • Description of needs of the Software Component Template toward AUTOSAR services and of the interaction of the Software Component Template and services (on XML level) • Legal disclaimer revised • Release notes added • "Advice for users" added • "Revision information" added
2.0.0	AUTOSAR Administration	Second
1.0.0	AUTOSAR Administration	Initial release

Disclaimer

This specification 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 specification.

The material contained in this specification is protected by copyright and other types of Intellectual Property Rights. The commercial exploitation of the material contained in this specification requires a license to such Intellectual Property Rights.

This specification 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 specification may be utilized or reproduced, in any form or by any means, without permission in writing from the publisher.

The AUTOSAR specifications have been developed for automotive applications only. They have neither been developed, nor tested for non-automotive applications.

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

Advice for users

AUTOSAR specifications may contain exemplary items (exemplary reference models, "use cases", and/or references to exemplary technical solutions, devices, processes or software).

Any such exemplary items are contained in the specifications for illustration purposes only, and they themselves are not part of the AUTOSAR Standard. Neither their presence in such specifications, nor any later documentation of AUTOSAR conformance of products actually implementing such exemplary items, imply that intellectual property rights covering such exemplary items are licensed under the same rules as applicable to the AUTOSAR Standard.

Table of Contents

1	Introduction	19
1.1	Overview	19
1.2	Scope	19
1.3	Organization of the Meta-Model	20
1.4	Structure of the Template	22
1.4.1	Description of Software Components on VFB Level	22
1.4.2	Description of Software Components on RTE Level	22
1.4.3	Descriptions of Software Components on Implementation Level	23
1.5	Abbreviations	23
1.6	Document Conventions	24
1.7	Requirements Tracing	26
2	Conceptual Aspects	35
2.1	Introduction	35
2.2	Measurement and Calibration	35
2.2.1	Basic Approach of Measurement and Calibration	35
2.2.2	Calibration Parameters Overview	35
2.2.3	Using Calibration Parameters	36
2.2.3.1	Sharing Calibration Parameters within Compositions	36
2.2.3.2	Sharing Calibration Parameters between SwComponentPrototypes of the Same SwComponentType	39
2.2.3.3	Providing Instance Individual Characteristic Data	40
2.3	Runtime and Data Consistency Aspects	41
2.3.1	Background: the Issues	41
2.3.1.1	Mutual Exclusion with Semaphores	42
2.3.1.2	Interrupt Disabling	42
2.3.1.3	Priority Ceiling	43
2.3.1.4	Implicit Communication by Means of Variable Copies	43
2.3.2	Data Consistency at Runtime	44
2.3.3	Modeling Aspects of Data Consistency	44
2.4	Variant Handling in the Software Component Template	45
2.5	Communication Specification of Composition Component Types	47
2.5.1	Rationale	47
2.6	PRPortPrototype	49
2.6.1	Use Case 1	49
2.6.2	Use Case 2	49
2.6.3	Use Case 3	50
2.6.4	Solution	51
2.7	Pretended Networking	51
2.8	Variable-size Array Data Types	53
2.8.1	Overview and Use cases	53
2.8.1.1	“Old-world” dynamic-size Arrays	53
2.8.1.2	“New-world” variable-size Arrays	54
2.8.2	Modeling Aspects regarding Application Data Types	56

2.8.3	Modeling Aspects regarding Implementation Data Types . . .	57
3	Overview: Software Components, Ports, and Interfaces	59
3.1	Introduction	59
3.2	Software Component	59
3.2.1	Overview	59
3.2.2	PortPrototype	61
3.2.3	AtomicSwComponentType	65
3.2.4	ParameterSwComponentType	67
3.2.5	Symbolic Name of a Software-Component	68
3.3	Composition	70
3.3.1	Overview	70
3.3.2	SwComponentPrototype	71
3.3.3	Connectors	75
3.3.4	Instantiation-specific RTEEvents	80
3.4	Port Interface	82
4	Details: Software Components, Ports, and Interfaces	88
4.1	Introduction	88
4.2	Port Interface Details	88
4.2.1	Introduction	88
4.2.2	Sender Receiver Communication	89
4.2.3	Client Server Communication	93
4.2.3.1	Client Server Interface	93
4.2.3.2	Error Handling in Client/Server Communication	100
4.2.4	External Trigger Event Communication	102
4.2.5	Communication of Modes	105
4.2.6	Parameter Communication	112
4.3	PortInterface Mapping and Data Scaling	112
4.3.1	PortInterface Mapping	115
4.3.1.1	Mapping of Sender Receiver Interface, Parameter In- terface and Non Volatile Data Interface Elements	116
4.3.1.2	Mapping of Client Server Interface Elements	119
4.3.1.3	Mapping of Mode Interface Elements	122
4.3.1.4	Mapping of Trigger Interface Elements	126
4.3.1.5	Mapping of Elements of a composite Data Type	127
4.3.2	Data Conversion	133
4.3.2.1	Linear Data Scaling	134
4.3.2.2	Table Conversion	135
4.4	Port Annotation	140
4.4.1	Introduction	140
4.4.2	SenderReceiverAnnotation	142
4.4.3	ClientServerAnnotation	146
4.4.4	Annotation for the I/O Hardware Abstraction Layer	147
4.4.5	Parameter Port Annotation	150
4.4.6	Mode Port Annotation	151
4.4.7	Trigger Port Annotation	152

4.4.8	Non Volatile Data Port Annotation	152
4.4.9	Delegated Port Annotations	153
4.4.10	General Annotation	155
4.5	Communication Specification	156
4.5.1	Communication Specification for Sender-Receiver Communication	159
4.5.2	Communication Specification for Client-Server Communication	170
4.5.3	Communication Specification for Mode Switch Communication	173
4.5.4	Communication Specification for Parameters	176
4.5.5	Communication Specification for NV Data	178
4.5.6	Configuration of Data Transformation	181
4.6	Port Groups within Component Types	187
4.7	End to End Protection	188
4.8	Partial Networking	200
4.8.1	VFC Control Ports	200
4.8.2	VFC Status Ports	201
4.9	Formal Definition of implicit Communication Behavior	202
4.9.1	Consistency Needs on Receiver Side	206
4.9.2	Consistency Needs on Sender Side	207
4.9.3	Consistency Needs for Senders and receivers of the same Data inside on RunnableEntityGroup	208
5	Data Description	209
5.1	Introduction	209
5.2	Data Types	213
5.2.1	Overview	213
5.2.2	Data Type Mapping	216
5.2.3	Data Categories	219
5.2.4	Application Data Type	223
5.2.4.1	Application Primitive Data Types	226
5.2.4.2	Application Composite Data Types	240
5.2.5	Implementation Data Type	252
5.2.6	Base Type	273
5.2.7	Data Type Terminology	279
5.2.7.1	Primitive Type	279
5.2.7.2	Compound Primitive Data Type	280
5.2.7.3	Integral Primitive Type	280
5.2.7.4	Variable-Size Array Data Type	282
5.3	Data Prototypes	282
5.3.1	Overview	282
5.3.2	Reference to Data Prototypes	289
5.4	Properties of Data Definitions	296
5.4.1	Overview	296
5.4.2	Invalid Value	311
5.4.3	Properties for Measurement	316
5.4.4	Properties of Curves and Maps	317

5.4.5	Setting an Axis Input Value	326
5.4.6	Specifying Data Dependencies	332
5.4.7	Precedence of data properties with respect to data elements, axis elements, computation methods, units	336
5.5	Elements used in Properties of Data Definitions	341
5.5.1	Computation Methods	341
5.5.1.1	Category Values in the context of a CompuMethod	351
5.5.1.2	Applicability of Attributes in the context of a CompuMethod	353
5.5.1.3	Example for Enumeration	355
5.5.1.4	Example for Linear Conversion	355
5.5.1.5	Example for Linear Conversion with texttable	356
5.5.1.6	Example for conversion specified by a rational function	357
5.5.1.7	Example for BITFIELD_TEXTTABLE	357
5.5.2	Physical Units, Physical Dimensions and Unit Groups	361
5.5.3	Data Constraints	368
5.5.4	Addressing Methods	375
5.5.5	Record Layouts	381
5.5.5.1	Specifying Record Layouts	382
5.5.5.2	RecordLayouts and DataTypes	392
5.5.5.3	Record Layouts and Interpolation Routines	398
5.6	Specification of Constant Values	401
5.6.1	Overview	401
5.6.2	Specification of Values based on Rules	406
5.6.3	Reference to Constant	414
5.6.4	Values for Compound Primitive Data Types	414
5.6.5	Examples	422
5.6.5.1	Example for Constant Specification for CURVE	422
5.6.5.2	Example for Constant Specification for MAP	423
5.6.5.3	Example for Constant Specification for COM_AXIS	424
5.7	Initial Values	425
5.7.1	Overview	425
5.7.2	Initial Value Representation	426
5.7.3	Constant Specification Mapping	427
5.7.4	Initial Values For CalibrationParameters	429
6	Compatibility	432
6.1	Introduction	432
6.2	Compatibility of Data Types	432
6.2.1	ApplicationDataType	432
6.2.1.1	ApplicationPrimitiveDataType	432
6.2.1.2	ApplicationCompositeDataType	433
6.2.2	ImplementationDataType	434
6.2.3	Compatibility of SwBaseType	435
6.2.4	Compatibility of SwDataDefProps	435
6.2.4.1	Compatibility of Units	436

6.2.4.2	Compatibility of PhysicalDimensions	437
6.2.4.3	Compatibility of Data Constraints	438
6.2.4.4	Compatibility in case of ImplementationDataType	438
6.2.4.5	Compatibility of CompuMethods	440
6.2.4.6	Compatibility of Record Layouts	441
6.2.5	Compatibility of ApplicationDataType and Implementation- DataType	442
6.3	Compatibility of Variable Data Prototypes and Parameter Data Prototypes	445
6.4	Compatibility of Sender Receiver Interfaces, Parameter Interfaces and Non Volatile Data Interfaces	446
6.4.1	Connection of Required and Provided Port via Assem- blySwConnector	447
6.4.2	Connection of Inner and Outer Port via DelegationSwCon- nector	447
6.4.3	Connection of Required and Provided Port via PassThrough- SwConnector	449
6.4.4	Compatibility of ParameterDataPrototype and VariableDat- aPrototype depending on PortInterface Type	449
6.5	Compatibility of Mode Switch Interfaces	450
6.5.1	Connection of Required and Provided Port via Assem- blySwConnector	450
6.5.2	Connection of Inner and Outer Port via DelegationSwCon- nector	451
6.5.3	Connection of Outer and Outer Port via PassThroughSwCon- nector	452
6.6	Compatibility of Mode Declaration Group Prototypes	452
6.7	Compatibility of Mode Declaration Groups	452
6.8	Compatibility of Argument Prototypes	454
6.9	Compatibility of Application Errors	454
6.10	Compatibility of Client/Server Operations	454
6.11	Compatibility of Client Server Interfaces	455
6.11.1	Connection of Required and Provided Port via Assem- blySwConnector	455
6.11.2	Connection of Inner and Outer Port via DelegationSwCon- nector	456
6.11.3	Connection of Outer and Outer Port via PassThroughSwCon- nector	456
6.12	Compatibility of Trigger Interfaces	457
6.12.1	Connection of Required and Provided Port via Assem- blySwConnector	457
6.12.2	Connection of Inner and Outer Port via DelegationSwCon- nector	457
6.12.3	Connection of Outer and Outer Port via PassThroughSwCon- nector	458
6.13	Compatibility of Trigger	458
6.14	Entire Delegation of a Provided Port Prototype	459

6.14.1	Split and Merge of PortInterface Elements	460
6.15	Compatibility in Case of a Flat ECU Extract	460
6.16	Compatibility Examples	461
6.16.1	Compatibility on Assembly Level	461
6.16.1.1	Legal Use	462
6.16.1.2	Illegal Use	462
6.16.2	Compatibility on Delegation Level	463
6.16.2.1	Legal Use	463
6.16.2.2	Illegal Use	467
7	Internal Behavior	470
7.1	Introduction	470
7.2	Runnable Entity	475
7.2.1	Concurrency and Reentrancy of a RunnableEntity that cannot be Invoked Concurrently	482
7.2.2	Concurrency and Reentrancy of a RunnableEntity that can be Invoked Concurrently	483
7.2.3	Timed Activation of Runnable Entities	484
7.2.4	Additional Remarks and Clarifications	485
7.2.4.1	Reentrancy and Multiple Instantiation	485
7.2.4.2	Reentrancy and “Library Functions”	486
7.2.4.3	Compatibility of ClientServerOperations triggering the same RunnableEntity	486
7.2.4.4	Categories of Runnable Entities	487
7.2.4.5	Arguments of a Runnable Entity	488
7.2.5	Activation Reason of a Runnable Entity	489
7.2.6	Runnable Entity for Initialization Purpose	491
7.3	RTEEvent	492
7.3.1	Defining an Event	498
7.3.2	Defining how to Respond to an Event	501
7.4	Communication among Runnable Entities	503
7.4.1	Description Possibility 1: Exclusive Area	504
7.4.1.1	Entire Runnable Runs in the Exclusive Area	507
7.4.1.2	Runnable would Dynamically Enter and Leave the Exclusive Area	508
7.4.2	Description Possibility 2: Inter-Runnable Variable	508
7.4.3	Inter Runnable Triggering	511
7.5	Data Access of RunnableEntities	512
7.5.1	RunnableEntities and Sender Receiver Communication	513
7.5.1.1	Terminology	513
7.5.1.2	Data Access	514
7.5.1.3	Explicit Sending and Receiving	518
7.5.1.4	DataSendCompletedEvent	521
7.5.1.5	DataWriteCompletedEvent	522
7.5.1.6	DataReceivedEvent	523
7.5.1.7	DataReceiveErrorEvent	524

7.5.2	RunnableEntities and Client Server Communication	526
7.5.2.1	Invoking an Operation	526
7.5.2.2	Providing an Implementation of an Operation	531
7.5.2.3	Reacting on Data Transformation Errors	533
7.5.3	RunnableEntities and External Trigger Event Communication	534
7.5.3.1	Trigger Source	534
7.5.3.2	Trigger Sink	535
7.5.4	RunnableEntities and Parameter Access	536
7.5.4.1	InstantiationDataDefProps	539
7.5.5	RunnableEntities and Mode Communication	541
7.6	Port API Options	542
7.6.1	Enable to Take Address	544
7.6.2	Indirect API Generation	544
7.6.3	Port Defined Argument Value	544
7.7	PerInstanceMemory	546
7.7.1	PerInstanceMemory typed by “C” Data Types	547
7.7.2	PerInstanceMemory typed by AUTOSAR Data Types	548
7.8	Static Memory and Constant Memory	549
7.9	Included AUTOSAR Data Types	550
7.10	Included Mode Declaration Groups	551
7.11	Service Needs	553
7.11.1	Overview	553
7.11.2	Assignment of Service Needs to Ports and Data	557
7.11.3	Specific Service Dependencies	566
7.11.3.1	NvM Service Dependencies	566
7.11.3.2	Watchdog Service Dependencies	573
7.11.3.3	COM Manager Service Needs	575
7.11.3.4	ECU State Manager Service Needs	576
7.11.3.5	BswM	580
7.11.3.6	Crypto Service Dependencies	583
7.11.3.7	Diagnostic Service Dependency	597
7.11.3.8	Diagnostic Log and Trace Dependency	647
7.11.3.9	Synchronized Time-Base Manager Dependency	648
7.12	Variation Point Proxy	650
8	Implementation	654
9	Mode Management	657
9.1	Declaration of Modes	657
9.2	Modes and Events	662
9.3	Initialization / Finalization	667
9.4	Mode Error Behavior	668
9.5	Summary Meta-Model Excerpt Related to Modes	672
10	ECU Abstraction and Complex Drivers	674
10.1	Introduction	674
10.2	High Level Hardware and Software Architecture	674

10.3	Interfaces and APIs	677
10.3.1	ECU Abstraction and its AUTOSAR Interfaces	678
10.4	Sensors/Actuators	678
10.5	I/O Hardware Abstraction	680
10.6	Complex Driver	682
11	Services	685
11.1	Overview: Generation of Service-related Model Elements	685
11.2	Extending the ECU Software Composition	688
11.3	Service Software Component Type	689
11.4	Service Proxy Component Type	692
11.5	Non Volatile Memory	695
11.5.1	Introduction	695
11.5.2	NvBlockComponent	695
11.5.3	Software-Components using <i>NVRAM data</i> of NvBlockComponents	697
11.5.4	NvBlockDescriptor	701
11.5.4.1	Writing Strategies	704
11.5.4.2	NvBlockNeeds	707
11.5.4.3	RAM Block and ROM Block	710
11.5.4.4	NvBlockDataMapping	711
11.5.4.5	Client Server Ports	716
11.5.5	SwcInternalBehavior of an NvBlockSwComponentType	718
12	Software Component Documentation	722
13	Rapid Prototyping Scenarios	726
13.1	Definition of Rapid Prototyping Scenario	726
13.2	Usage of RptContainers on M1	729
13.3	Usage of atpSplittable for RptContainers on M1	730
13.4	Modifications of the Meta-Model for supporting the RPT scenario	731
A	Renamed Meta-Model Elements	735
A.1	Introduction	735
A.2	Renamed Meta-Model Elements	735
B	Glossary	737
C	History of Constraints and Specification Items	740
C.1	Constraint History of this Document according to AUTOSAR R4.0.1	740
C.1.1	Changed Constraints in R4.0.1	740
C.1.2	Added Constraints in R4.0.1	740
C.1.3	Deleted Constraints	744
C.2	Constraint History of this Document according to AUTOSAR R4.0.2	744
C.2.1	Changed Constraints in R4.0.2	744
C.2.2	Added Constraints in R4.0.2	744
C.2.3	Deleted Constraints in R4.0.2	745

C.3	Constraint History of this Document according to AUTOSAR R4.0.3 . . .	746
C.3.1	Changed Constraints in R4.0.3	746
C.3.2	Added Constraints in R4.0.3	746
C.3.3	Added Specification Items in R4.0.3	748
C.3.4	Deleted Constraints in R4.0.3	760
C.3.5	Deleted Specification Items	760
C.4	Constraint History of this Document according to AUTOSAR R4.1.1 . .	761
C.4.1	Changed Constraints in R4.1.1	761
C.4.2	Added Constraints in R4.1.1	762
C.4.3	Changed Specification Items in R4.1.1	764
C.4.4	Added Specification Items in R4.1.1	764
C.4.5	Deleted Constraints in R4.1.1	767
C.4.6	Deleted Specification Items in R4.1.1	767
C.5	Constraint History of this Document according to AUTOSAR R4.1.2 . .	767
C.5.1	Changed Constraints in R4.1.2	767
C.5.2	Added Constraints in R4.1.2	768
C.5.3	Changed Specification Items in R4.1.2	769
C.5.4	Added Specification Items in R4.1.2	769
C.5.5	Deleted Constraints in R4.1.2	769
C.5.6	Deleted Specification Items in R4.1.2	770
C.6	Constraint History of this Document according to AUTOSAR R4.1.3 . .	770
C.6.1	Added Traceables in 4.1.3	770
C.6.2	Changed Traceables in 4.1.3	770
C.6.3	Deleted Traceables in 4.1.3	770
C.6.4	Added Constraints in 4.1.3	771
C.6.5	Changed Constraints in 4.1.3	771
C.6.6	Deleted Constraints in 4.1.3	771
C.7	Constraint History of this Document according to AUTOSAR R4.2.1 . .	771
C.7.1	Added Traceables in 4.2.1	771
C.7.2	Changed Traceables in 4.2.1	773
C.7.3	Deleted Traceables in 4.2.1	774
C.7.4	Added Constraints in 4.2.1	774
C.7.5	Changed Constraints in 4.2.1	775
C.7.6	Deleted Constraints in 4.2.1	775
C.8	Constraint History of this Document according to AUTOSAR R4.2.2 . .	776
C.8.1	Added Traceables in 4.2.2	776
C.8.2	Changed Traceables in 4.2.2	777
C.8.3	Deleted Traceables in 4.2.2	777
C.8.4	Added Constraints in 4.2.2	777
C.8.5	Changed Constraints in 4.2.2	778
C.8.6	Deleted Constraints in 4.2.2	779
D	Modeling of InstanceRef	780
D.1	Introduction	780
D.2	Modeling	781
D.2.1	Components and Compositions	781

D.2.2	Definition of implicit Communication Behavior	799
D.2.3	Internal Behavior	809
E	Examples	819
E.1	Examples for the Definition of variable-size Arrays	819
F	Mentioned Class Tables	823
G	Upstream Mapping	876
G.1	Introduction	876
G.2	NvM	876
G.3	Com	883
G.4	WdgM	897
G.5	Dcm	897
H	Splitable Elements in the Scope of this Document	901
I	Variation Points in the Scope of this Document	903

Bibliography

- [1] Standardization Template
AUTOSAR_TPS_StandardizationTemplate
- [2] Specification of RTE Software
AUTOSAR_SWS_RTE
- [3] Virtual Functional Bus
AUTOSAR_EXP_VFB
- [4] Methodology
AUTOSAR_TR_Methodology
- [5] Interoperability of AUTOSAR Tools
AUTOSAR_TR_InteroperabilityOfAutosarTools
- [6] Layered Software Architecture
AUTOSAR_EXP_LayeredSoftwareArchitecture
- [7] Basic Software Module Description Template
AUTOSAR_TPS_BSWModuleDescriptionTemplate
- [8] Specification of Timing Extensions
AUTOSAR_TPS_TimingExtensions
- [9] Requirements on Timing Extensions
AUTOSAR_RS_TimingExtensions
- [10] Specification of ECU Resource Template
AUTOSAR_TPS_ECUResourceTemplate
- [11] System Template
AUTOSAR_TPS_SystemTemplate
- [12] Generic Structure Template
AUTOSAR_TPS_GenericStructureTemplate
- [13] Requirements on Software Component Template
AUTOSAR_RS_SoftwareComponentTemplate
- [14] Supplementary material of general blueprints for AUTOSAR
AUTOSAR_TR_GeneralBlueprintsSupplement
- [15] Specification of Basic Software Mode Manager
AUTOSAR_SWS_BSWModeManager
- [16] Information technology – Universal Coded Character Set (UCS)
<http://www.iso.org>
- [17] Specification of I/O Hardware Abstraction
AUTOSAR_SWS_IOHardwareAbstraction

- [18] Communication
<http://portal.osek-vdx.org/files/pdf/specs/osekcom303.pdf>
- [19] Specification of SW-C End-to-End Communication Protection Library
AUTOSAR_SWS_E2ELibrary
- [20] Specification of Communication Manager
AUTOSAR_SWS_COMManager
- [21] Specification of Communication
AUTOSAR_SWS_COM
- [22] Specification of Platform Types
AUTOSAR_SWS_PlatformTypes
- [23] ASAM MCD 2 Harmonized Data Objects Version 1.1
[harmonized-data-objects-V1.1.pdf](#)
- [24] XML Specification of Application Interfaces
AUTOSAR_MOD_AISpecification
- [25] ASAM MCD 2MC ASAP2 Interface Specification
<http://www.asam.net>
[ASAP2-V1.51.pdf](#)
- [26] ISO 26262 (Part 1-10) – Road vehicles – Functional Safety, First edition
<http://www.iso.org>
- [27] ASAM AE Calibration Data Format V2.0.0
<http://www.asam.net>
[ASAM-AE-CDF-V2_0_0-Users-Guide.pdf](#)
- [28] Specification of Operating System
AUTOSAR_SWS_OS
- [29] Specification of ECU Configuration Parameters (XML)
AUTOSAR_MOD_ECUConfigurationParameters
- [30] Collection of blueprints for AUTOSAR M1 models
AUTOSAR_MOD_GeneralBlueprints
- [31] Specification of NVRAM Manager
AUTOSAR_SWS_NVRAMManager
- [32] Specification of Watchdog Manager
AUTOSAR_SWS_WatchdogManager
- [33] Specification of ECU State Manager
AUTOSAR_SWS_ECUCStateManager
- [34] Specification of Crypto Service Manager
AUTOSAR_SWS_CryptoServiceManager

- [35] Specification of Function Inhibition Manager
AUTOSAR_SWS_FunctionInhibitionManager
- [36] Specification of Diagnostic Event Manager
AUTOSAR_SWS_DiagnosticEventManager
- [37] Specification of Diagnostic Communication Manager
AUTOSAR_SWS_DiagnosticCommunicationManager
- [38] Diagnostic Extract Template
AUTOSAR_TPS_DiagnosticExtractTemplate
- [39] Road vehicles – Diagnostic communication over Internet Protocol (DoIP)
<http://www.iso.org>
- [40] Specification of Diagnostic Log and Trace
AUTOSAR_SWS_DiagnosticLogAndTrace
- [41] Specification of Synchronized Time-Base Manager
AUTOSAR_SWS_SynchronizedTimeBaseManager
- [42] Glossary
AUTOSAR_TR_Glossary
- [43] ASAM AE Functional Specification Exchange Format V1.0.0
<http://www.asam.net>
AE-FSX_V1.0.0.pdf
- [44] Software Process Engineering Meta-Model Specification
<http://www.omg.org/spec/SPEM/2.0/>

1 Introduction

1.1 Overview

This document contains the specification of the AUTOSAR Software-Component Template. Actually, it has been created as a supplement to the formal definition of the Software-Component Template by means of the AUTOSAR meta-model. In other words, this document in addition to the formal specification provides introductory description and rationale for the part of the AUTOSAR meta-model relevant for the definition of software-components.

In this context, the term software-component refers to a formally described piece of software existing that needs the AUTOSAR RTE [2] for execution.

Please note that the general ideas behind the semantics of application software-components have been described in the specification of the Virtual Functional Bus [3]. The latter, however, represents conceptual work that strongly influences but does not totally govern the formal definition of software-components.

Note further that this document does not provide any “best practice” recommendations of software-component modeling nor does it require or enforce a certain methodology. Note however, that the methodology aspect is covered by the specification of the AUTOSAR methodology [4].

Although it is beyond any doubt reasonable to use a suitable AUTOSAR Authoring Tool for dealing with AUTOSAR software-components, this specification does not make any assumptions nor does it give recommendations regarding the tooling. Please refer to [5] for more details about AUTOSAR Authoring Tools are supposed to work and interact.

1.2 Scope

As already mentioned in chapter 1.1, the Scope of this document is the description of AUTOSAR software-components. This work covers the following three aspects:

- A general description of `SwComponentTypes` using `PortPrototypes` and `PortInterfaces`, i.e. this document defines the `SwComponentType` as an entity which can be described through `PortPrototypes` which provide or require `PortInterfaces`.
- A description of `CompositionSwComponentTypes` which are sub-systems consisting out of connected instances of software-components, i.e. software-components may be defined in the form of hierarchical subsystems which in turn consist of software-components again. The description of such hierarchical structures is in scope of this document.

- A description of `AtomicSwComponentType` which is implemented as a piece of software that can be mapped to an AUTOSAR ECU.

An `AtomicSwComponentType` therefore shows up in the ECU Software Architecture depicted in Figure 1.1. In this figure, the green (vertically striped) and blue (diagonally striped) borders show the aspects that are described by the Software-Component Template.

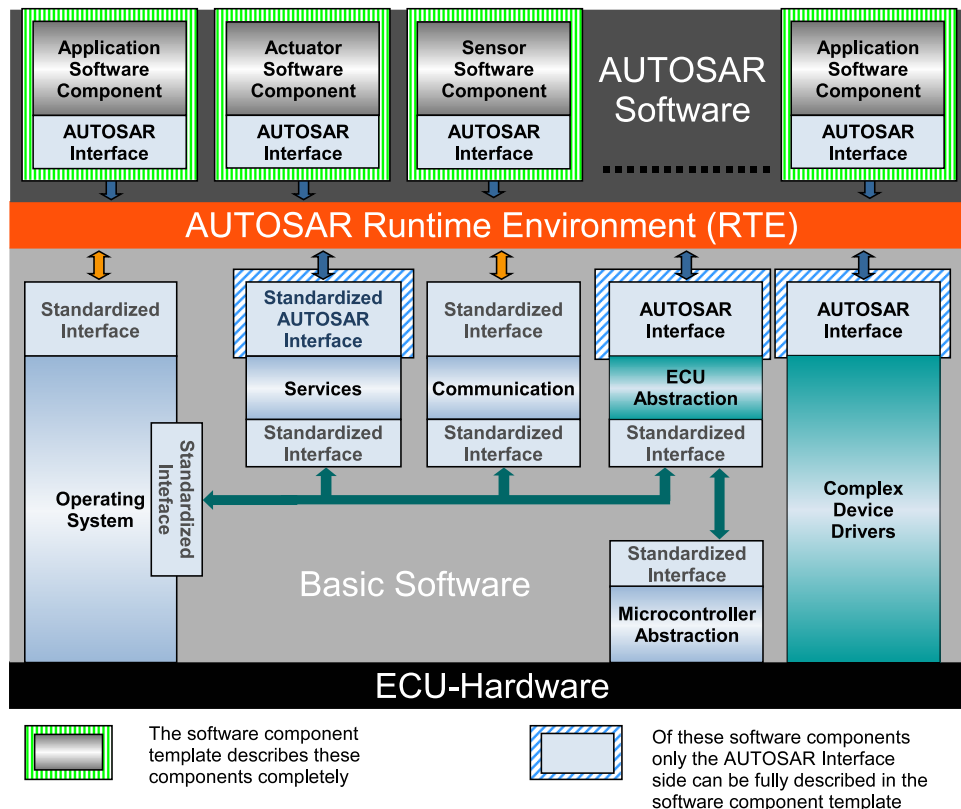


Figure 1.1: Scope of this document in the ECU SW Architecture [6]

Aspects of AUTOSAR Basic Software not relevant for the RTE are out of scope; these are covered by the Basic Software Module Description Template [7].

Also, the document does not cover aspects of timing analysis with respect to the execution of AUTOSAR software-components. This issue is explained in the Specification of Timing Extensions [8] as well as the corresponding requirements specification [9].

1.3 Organization of the Meta-Model

Figure 1.2 sketches the overall structure of the meta-model which formally defines the vocabulary required to describe AUTOSAR software-components. As the diagram points out, other template specifications (e.g. ECU Resource Template [10] and System Template [11]) also use the same modeling approach in order to define an overall consistent model of AUTOSAR software description.

The dashed arrows in the diagram describe dependencies in terms of import-relationships between the packages within the meta-model. For example, the package `SWComponentTemplate` imports meta-classes defined in the packages `GenericStructure` [12] and `ECUResourceTemplate` [10].

Please note that this specification document will (with some well-defined exceptions) mostly discuss meta-model elements defined in the package `SWComponentTemplate`.

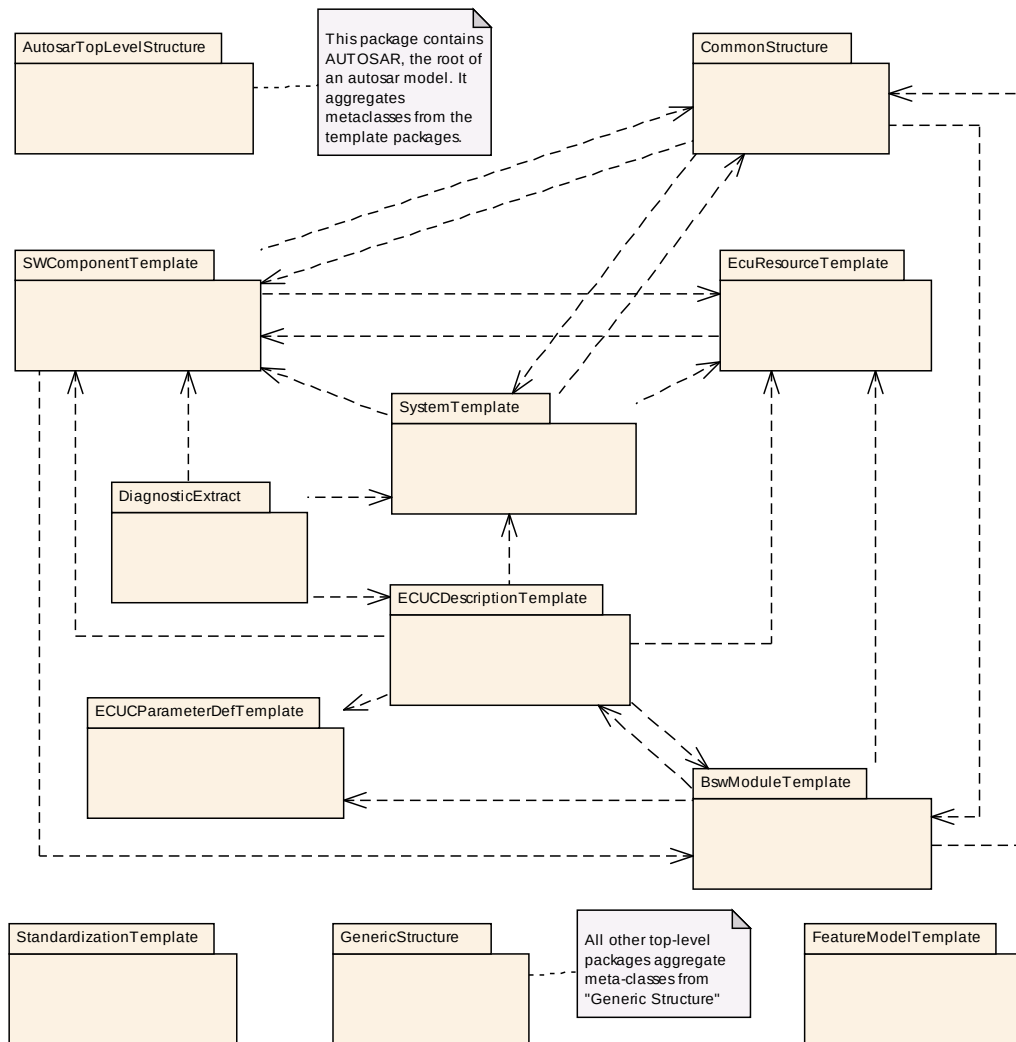


Figure 1.2: Structure of the meta-model

For clarification, please note that the package `GenericStructure` contains some fundamental infrastructure meta-classes and common patterns that are described in [12]. As these are used by all other template specification the dependency associations are not depicted in the diagram for the sake of clarity.

1.4 Structure of the Template

AUTOSAR software components are described on three distinctive levels, as shown in Figure 1.3.

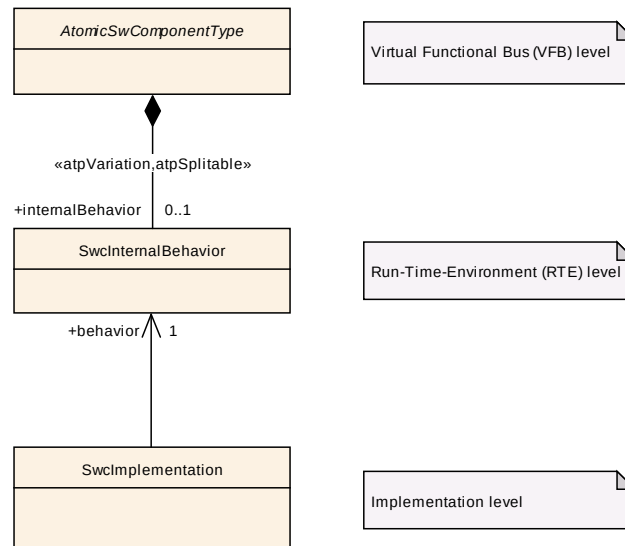


Figure 1.3: The description of a software component is done on three levels

1.4.1 Description of Software Components on VFB Level

The highest (most abstract) description level is the Virtual Functional Bus [3]. In this document `SwComponentTypes` are described with the means of `DataTypes`, `PortInterfaces`, `PortPrototypes`, and connections between them. At this level, the fundamental communication properties of components and their communication relationships among each other are expressed.

In the diagram depicted in Figure 1.3, this aspect is expressed by means of the description of `AtomicSwComponentType`¹.

1.4.2 Description of Software Components on RTE Level

The middle level allows for behavior description of a given `AtomicSwComponentType`. This so-called `SwcInternalBehavior` is expressed according to AUTOSAR RTE concepts, e.g. `RTEEvents` and in terms of schedulable units, so-called `RunnableEntities`.

For instance, for a `ClientServerOperation` defined in the scope of a particular `ClientServerInterface` on the VFB, the behavior specifies which `RunnableEn-`

¹To avoid clutter and require additional up-front information about the meta model, `Composition-SwComponentTypes` have not been added to the diagram.

`tity` is activated as a consequence of the invocation of the specific `ClientServer-Operation`.

As sketched by Figure 1.3, there may be zero or one `SwcInternalBehaviors` aggregated by a given `AtomicSwComponentType`. In response to the existence of the stereotype `<<atpSplitable>>` at the aggregation it is possible to distribute the aggregation over several physical files.

1.4.3 Descriptions of Software Components on Implementation Level

The lowest level of description specifies the implementation (i.e. in terms of the AUTOSAR meta-model: the `SwcImplementation`) of a given `SwcInternalBehavior` description. More precisely, the `RunnableEntitys` of such a behavior are mapped to code (source code or object code).

There may be different `SwcImplementations` that reference a specific `SwcInternalBehavior` description, e.g. in different programming languages, or with differently optimized code.

Please note that `Implementation` has been described in previous versions of this document. In response to the evolution of the AUTOSAR concept the description of the `Implementation` aspect has been moved to the “CommonStructure” (see Figure 1.2) because it is also used for creating the Basic Software Module Description Template [7].

However, the `SwcImplementation` still remains in the scope of this document as it exclusively covers aspects of software-components rather than basic software modules.

1.5 Abbreviations

The following table contains a list of abbreviations used in the scope of this document along with the spelled-out meaning of each of the abbreviations.

<i>Abbreviation</i>	<i>meaning</i>
API	Application Programming Interface
BOM	Byte Order Mark
CAN	Controller Area Network
CSE	Codes for Scaling Units
DCM	Diagnostics Communication Manager
DCY	Driving Cycle
DEM	Diagnostics Event Manager
DID	Diagnostic Identifier
DTC	Diagnostic Trouble Code
Dolp	Diagnostics over IP
ECU	Electrical Control Unit
EPROM	Erasable Programmable Read-Only Memory

EEPROM	Electrically Erasable Programmable Read-Only Memory
FID	Function Identifier
GID	Group Identifier
ID	Identifier
IO	Input/Output
IP	Internet Protocol
IUMPR	In-Use Monitor Performance Ratio
ISO	International Standardization Organization
MAC	Message Authentication Code
MCAL	Micro-Controller Abstraction
LIN	Local Interconnect Network
MCD	Measurement, Calibration, Diagnostics
NM	Network Management
NV	Non-Volatile
OBD	On-Board Diagnostic
OEM	Original Equipment Manufacturer
OS	Operating System
PDU	Protocol Data Unit
PID	Parameter Identifier
PTO	Power Take Off
RA	Routing Activation
RAM	Random Access Memory
ROM	Read-Only Memory
RPT	Rapid Prototyping
RTE	Runtime Environment
SWC	Software Component
TID	Test Identifier
UDS	Unified Diagnostic Services
UML	Unified Modeling Language
VFB	Virtual Functional Bus
WWH-OBD	World-Wide Harmonized On-Board Diagnostics
XML	Extensible Markup Language
XSD	XML Schema Definition

Table 1.1: Abbreviations used in the scope of this Document

1.6 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. `PortPrototypes`. 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:

Class	AUTOSAR			
Package	M2::AUTOSARTemplates::AutosarTopLevelStructure			
Note	Root element of an AUTOSAR description, also the root element in corresponding XML documents. Tags: xml.globalElement=true			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
adminData	AdminData	0..1	aggr	This represents the administrative data of an Autosar file. Tags: xml.sequenceOffset=10
arPackage	ARPackage	*	aggr	This is the top level package in an AUTOSAR model. Stereotypes: atpSplitable; atpVariation Tags: atp.Splitkey=shortName, variation Point.shortLabel vh.latestBindingTime=blueprintDerivationTime xml.sequenceOffset=30
introduction	Documentation Block	0..1	aggr	This represents an introduction on the Autosar file. It is intended for example to represent disclaimers and legal notes. Tags: xml.sequenceOffset=20

Table 1.2: AUTOSAR

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.

Datatype: The datatype 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 attributes is aggregated in the class (*aggr*), an UML attribute in the class (*attr*), or just referenced by it (*ref*). Instance references are also indicated (*iref*) 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.

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

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

1.7 Requirements Tracing

Requirements against this document are exclusively stated in the corresponding requirements document [13].

The following table references the requirements specified in [13] and provides information about individual specification items that fulfill a given requirement.

Requirement	Description	Satisfied by
[RS_SWCT_00010]	AUTOSAR shall support inter- and intra-ECU-communication mechanisms with high reliability	[TPS_SWCT_01025] [TPS_SWCT_01026] [TPS_SWCT_01027] [TPS_SWCT_01069] [TPS_SWCT_01070] [TPS_SWCT_01111] [TPS_SWCT_01516] [TPS_SWCT_01573]
[RS_SWCT_00020]	AUTOSAR shall provide open and standardized software interfaces for intra-ECU and inter-ECU communication	[TPS_SWCT_01002]
[RS_SWCT_00030]	AUTOSAR shall provide complete interfaces to application software and basic software modules	[TPS_SWCT_01002]
[RS_SWCT_00070]	AUTOSAR shall provide an abstraction of the application software from hardware	[TPS_SWCT_01030] [TPS_SWCT_01097] [TPS_SWCT_01098]
[RS_SWCT_00080]	AUTOSAR shall provide an independence of application software from in-vehicle communication technologies	[TPS_SWCT_01025] [TPS_SWCT_01026] [TPS_SWCT_01027] [TPS_SWCT_01069] [TPS_SWCT_01070] [TPS_SWCT_01516]
[RS_SWCT_00090]	AUTOSAR should provide an independence of application software from operating systems	[TPS_SWCT_01030] [TPS_SWCT_01097] [TPS_SWCT_01098]

[RS_SWCT_00110]	AUTOSAR shall provide a functional interface view of the entire system	[TPS_SWCT_01025]	[TPS_SWCT_01026]
		[TPS_SWCT_01027]	[TPS_SWCT_01069]
		[TPS_SWCT_01070]	[TPS_SWCT_01516]
[RS_SWCT_00120]	AUTOSAR shall provide protection/unlock mechanisms for software through appropriate services in the infrastructure	[TPS_SWCT_01031]	[TPS_SWCT_01049]
		[TPS_SWCT_01050]	[TPS_SWCT_01051]
		[TPS_SWCT_01052]	[TPS_SWCT_01053]
		[TPS_SWCT_01054]	[TPS_SWCT_01055]
		[TPS_SWCT_01321]	[TPS_SWCT_01592]
[RS_SWCT_00150]	AUTOSAR shall provide means to protect SW-Components from malicious SW-Components	[TPS_SWCT_01002]	
[RS_SWCT_00160]	AUTOSAR shall provide means to achieve compositionality	[TPS_SWCT_01002]	
[RS_SWCT_00170]	AUTOSAR shall provide diagnostics means during runtime, for production and services purposes	[TPS_SWCT_01028]	[TPS_SWCT_01029]
		[TPS_SWCT_01132]	[TPS_SWCT_01133]
		[TPS_SWCT_01134]	[TPS_SWCT_01135]
		[TPS_SWCT_01136]	[TPS_SWCT_01137]
		[TPS_SWCT_01138]	[TPS_SWCT_01139]
		[TPS_SWCT_01140]	[TPS_SWCT_01425]
		[TPS_SWCT_01426]	[TPS_SWCT_01427]
		[TPS_SWCT_01428]	[TPS_SWCT_01453]
		[TPS_SWCT_01582]	[TPS_SWCT_01627]
		[TPS_SWCT_01628]	[TPS_SWCT_01629]
		[TPS_SWCT_01630]	[TPS_SWCT_01631]
		[TPS_SWCT_01632]	[TPS_SWCT_01633]
		[TPS_SWCT_01634]	[TPS_SWCT_01639]
		[TPS_SWCT_01640]	[TPS_SWCT_01654]
		[TPS_SWCT_01655]	[TPS_SWCT_01656]
		[TPS_SWCT_01657]	[TPS_SWCT_01658]
		[TPS_SWCT_02002]	[TPS_SWCT_02003]
		[TPS_SWCT_02004]	[TPS_SWCT_02005]
		[TPS_SWCT_02007]	[TPS_SWCT_02008]
		[TPS_SWCT_02009]	[TPS_SWCT_02010]
		[TPS_SWCT_02011]	[TPS_SWCT_02012]
		[TPS_SWCT_02013]	[TPS_SWCT_02014]
		[TPS_SWCT_02015]	[TPS_SWCT_02016]
[RS_SWCT_00190]	AUTOSAR shall support hierarchical design methods	[TPS_SWCT_01032]	[TPS_SWCT_01033]
		[TPS_SWCT_01034]	[TPS_SWCT_01035]
		[TPS_SWCT_01036]	[TPS_SWCT_01037]
[RS_SWCT_00200]	Definitions of relations between SW components are exhaustive and formal	[TPS_SWCT_01002]	[TPS_SWCT_01322]
		[TPS_SWCT_01323]	[TPS_SWCT_01325]
		[TPS_SWCT_01326]	[TPS_SWCT_01328]
		[TPS_SWCT_01329]	[TPS_SWCT_01330]
		[TPS_SWCT_01331]	[TPS_SWCT_01333]
		[TPS_SWCT_01334]	[TPS_SWCT_01335]
		[TPS_SWCT_01336]	[TPS_SWCT_01337]
		[TPS_SWCT_01338]	[TPS_SWCT_01339]
		[TPS_SWCT_01340]	[TPS_SWCT_01341]
		[TPS_SWCT_01342]	[TPS_SWCT_01343]
		[TPS_SWCT_01344]	[TPS_SWCT_01345]
		[TPS_SWCT_01346]	[TPS_SWCT_01347]

		[TPS_SWCT_01348] [TPS_SWCT_01349] [TPS_SWCT_01350] [TPS_SWCT_01351] [TPS_SWCT_01352] [TPS_SWCT_01353] [TPS_SWCT_01557] [TPS_SWCT_01558] [TPS_SWCT_01567]
[RS_SWCT_00210]	SW components are protected from illegal access	[TPS_SWCT_01002]
[RS_SWCT_00220]	Management of vehicle diversity is supported by AUTOSAR	[TPS_SWCT_01038] [TPS_SWCT_01039] [TPS_SWCT_01040] [TPS_SWCT_01041] [TPS_SWCT_01042] [TPS_SWCT_01447]
[RS_SWCT_00230]	The Software Component Template shall provide the ability to define naming conventions for public symbols	[TPS_SWCT_01635]
[RS_SWCT_02000]	AUTOSAR shall support a top-down hierarchical design	[TPS_SWCT_01032] [TPS_SWCT_01033] [TPS_SWCT_01034] [TPS_SWCT_01035] [TPS_SWCT_01036] [TPS_SWCT_01037]
[RS_SWCT_02010]	Interfaces of atomic software-components shall be supported	[TPS_SWCT_01002]
[RS_SWCT_02020]	Bottom-up design of CompositionTypes shall be supported	[TPS_SWCT_01032] [TPS_SWCT_01033] [TPS_SWCT_01034] [TPS_SWCT_01035] [TPS_SWCT_01036] [TPS_SWCT_01037]
[RS_SWCT_02030]	Specification of Communications shall be supported	[TPS_SWCT_01002] [TPS_SWCT_01025] [TPS_SWCT_01026] [TPS_SWCT_01027] [TPS_SWCT_01516]
[RS_SWCT_02060]	Interaction with basic software shall be considered	[TPS_SWCT_01043] [TPS_SWCT_01044] [TPS_SWCT_01045] [TPS_SWCT_01046]
[RS_SWCT_02080]	Designing a Sensor Actuator Component shall be supported	[TPS_SWCT_01047] [TPS_SWCT_01048]
[RS_SWCT_02090]	Data-consistency for communication among RunnableEntities shall be supported	[TPS_SWCT_01031] [TPS_SWCT_01049] [TPS_SWCT_01050] [TPS_SWCT_01051] [TPS_SWCT_01052] [TPS_SWCT_01053] [TPS_SWCT_01054] [TPS_SWCT_01055] [TPS_SWCT_01637]
[RS_SWCT_02100]	Definition of physical units shall be supported	[TPS_SWCT_01056] [TPS_SWCT_01057] [TPS_SWCT_01058] [TPS_SWCT_01059] [TPS_SWCT_01060] [TPS_SWCT_01061] [TPS_SWCT_01068]
[RS_SWCT_02110]	Definition of comments shall be supported	[TPS_SWCT_01062]
[RS_SWCT_03000]	The SW-Component template shall support compositions	[TPS_SWCT_01032] [TPS_SWCT_01033] [TPS_SWCT_01034] [TPS_SWCT_01035] [TPS_SWCT_01036] [TPS_SWCT_01037]
[RS_SWCT_03010]	The SW-Component template shall support interfaces	[TPS_SWCT_01025] [TPS_SWCT_01026] [TPS_SWCT_01069] [TPS_SWCT_01070] [TPS_SWCT_01516]
[RS_SWCT_03040]	The SW-Component template shall support description of the behavior	[TPS_SWCT_01075] [TPS_SWCT_01108]

[RS_SWCT_03045]	The SW-Component template shall allow enabling of RTE-Feature to get the activating RTE-Event of Runnable Entity	[TPS_SWCT_01469]
[RS_SWCT_03046]	The SW-Component template shall support instance specific RTE-Events	[TPS_SWCT_02507]
[RS_SWCT_03050]	The SW-Component template shall support the definition of schedulability	[TPS_SWCT_01030] [TPS_SWCT_01097] [TPS_SWCT_01098]
[RS_SWCT_03055]	The SW-Component template shall support optional configuration of ExclusiveArea usage within RunnableEntities	[TPS_SWCT_01457] [TPS_SWCT_01458] [TPS_SWCT_01459] [TPS_SWCT_01460]
[RS_SWCT_03065]	The SW-Component template shall support the definition of implicit communication behavior	[TPS_SWCT_01466] [TPS_SWCT_01470] [TPS_SWCT_01471] [TPS_SWCT_01472] [TPS_SWCT_01473] [TPS_SWCT_01475] [TPS_SWCT_01476] [TPS_SWCT_01479] [TPS_SWCT_01481] [TPS_SWCT_01482] [TPS_SWCT_01509] [TPS_SWCT_01625]
[RS_SWCT_03090]	The SW-Component template shall support the definition of needed and usable sensors and actuators	[TPS_SWCT_01047] [TPS_SWCT_01048]
[RS_SWCT_03100]	The SW-Component template shall support variant handling	[TPS_SWCT_01038] [TPS_SWCT_01040] [TPS_SWCT_01041] [TPS_SWCT_01042] [TPS_SWCT_01370] [TPS_SWCT_01371] [TPS_SWCT_01372] [TPS_SWCT_01373] [TPS_SWCT_01448]
[RS_SWCT_03110]	The SW-Component template shall support modes	[TPS_SWCT_01071] [TPS_SWCT_01190] [TPS_SWCT_01376] [TPS_SWCT_01377] [TPS_SWCT_01378] [TPS_SWCT_01379] [TPS_SWCT_01380] [TPS_SWCT_01381] [TPS_SWCT_01382] [TPS_SWCT_01383] [TPS_SWCT_01384] [TPS_SWCT_01385] [TPS_SWCT_01388] [TPS_SWCT_01511] [TPS_SWCT_01512] [TPS_SWCT_01513] [TPS_SWCT_01530] [TPS_SWCT_01531] [TPS_SWCT_01532] [TPS_SWCT_01533] [TPS_SWCT_01534] [TPS_SWCT_01535] [TPS_SWCT_01536] [TPS_SWCT_01541] [TPS_SWCT_01542] [TPS_SWCT_01552] [TPS_SWCT_01553] [TPS_SWCT_01554] [TPS_SWCT_01555] [TPS_SWCT_01581]
[RS_SWCT_03115]	The SW-Component template shall support mapping of mode declarations	[TPS_SWCT_01464] [TPS_SWCT_01465] [TPS_SWCT_01545]
[RS_SWCT_03120]	The SW-Component template shall support dependency on modes	[TPS_SWCT_01077]

[RS_SWCT_03130]	The SW-Component template shall support connections between PortInterfaces	[TPS_SWCT_01079] [TPS_SWCT_01080] [TPS_SWCT_01081] [TPS_SWCT_01082] [TPS_SWCT_01083] [TPS_SWCT_01084] [TPS_SWCT_01113] [TPS_SWCT_01573]
[RS_SWCT_03135]	The SW-Component template shall support record type subsetting	[TPS_SWCT_01023] [TPS_SWCT_01024] [TPS_SWCT_01551]
[RS_SWCT_03136]	The SW-Component template shall support record type subsetting with primitive types	[TPS_SWCT_01195]
[RS_SWCT_03140]	The SW-Component template shall support conditional existence of PortPrototypes	[TPS_SWCT_01038]
[RS_SWCT_03141]	The SW-Component template shall support the conditional existence of data element prototypes, operation prototypes, parameter prototypes in an interface	[TPS_SWCT_01106]
[RS_SWCT_03142]	The SW-Component template shall support the conditional existence of Component Prototypes	[TPS_SWCT_01038]
[RS_SWCT_03143]	The SW-Component template shall support the conditional existence of Connector Prototypes	[TPS_SWCT_01040]
[RS_SWCT_03144]	The SW-Component template shall support a configurable size of arrays	[TPS_SWCT_01076] [TPS_SWCT_01078]
[RS_SWCT_03148]	Attributes swMinAxisPoints and swMaxAxisPoints shall be adjustable by an System Constant Definition	[TPS_SWCT_01107] [TPS_SWCT_01181]
[RS_SWCT_03149]	The SW-Component template shall support the conditional existence of RunnableEntitys	[TPS_SWCT_01085]
[RS_SWCT_03150]	The SW-Component template shall support the conditional existence of RTEEvents	[TPS_SWCT_01085]
[RS_SWCT_03151]	The SW-Component template shall support the conditional existence of InterRunnable Variables	[TPS_SWCT_01085]
[RS_SWCT_03152]	The SW-Component template shall support the conditional accessibility for measurement	[TPS_SWCT_01130]
[RS_SWCT_03153]	The SW-Component template shall support the conditional existence of parameter prototypes	[TPS_SWCT_01085]
[RS_SWCT_03154]	The SW-Component template shall support conditional ports for software components	[TPS_SWCT_01038]

[RS_SWCT_03155]	The SW-Component template shall support interfaces with different resolutions	[TPS_SWCT_01099] [TPS_SWCT_01101] [TPS_SWCT_01103] [TPS_SWCT_01105]	[TPS_SWCT_01100] [TPS_SWCT_01102] [TPS_SWCT_01104]
[RS_SWCT_03170]	The SW-Component template shall support fixed data exchange	[TPS_SWCT_01102] [TPS_SWCT_01104]	[TPS_SWCT_01103]
[RS_SWCT_03175]	The SW-Component template shall support the definition of calibration datasets	[TPS_SWCT_01177] [TPS_SWCT_01188]	[TPS_SWCT_01178]
[RS_SWCT_03180]	The SW-Component template shall support SAE J1939 Protocol Features	[TPS_SWCT_01076]	
[RS_SWCT_03181]	The SW-Component template shall support arrays of variable number of elements within the maximum size	[TPS_SWCT_01076] [TPS_SWCT_01495] [TPS_SWCT_01602] [TPS_SWCT_01605] [TPS_SWCT_01607] [TPS_SWCT_01610] [TPS_SWCT_01613] [TPS_SWCT_01615] [TPS_SWCT_01618] [TPS_SWCT_01620] [TPS_SWCT_01622] [TPS_SWCT_01636] [TPS_SWCT_01642] [TPS_SWCT_01645] [TPS_SWCT_01648] [TPS_SWCT_01650]	[TPS_SWCT_01127] [TPS_SWCT_01601] [TPS_SWCT_01604] [TPS_SWCT_01606] [TPS_SWCT_01608] [TPS_SWCT_01612] [TPS_SWCT_01614] [TPS_SWCT_01617] [TPS_SWCT_01619] [TPS_SWCT_01621] [TPS_SWCT_01623] [TPS_SWCT_01641] [TPS_SWCT_01644] [TPS_SWCT_01647] [TPS_SWCT_01649]
[RS_SWCT_03182]	The SW-Component template shall support byte arrays of variable number of elements	[TPS_SWCT_01127]	
[RS_SWCT_03190]	The SW-Component template shall support the ability to publish/specify the diagnostic capabilities and its resources of an SWC	[TPS_SWCT_01028] [TPS_SWCT_01129] [TPS_SWCT_01133] [TPS_SWCT_01135] [TPS_SWCT_01137] [TPS_SWCT_01139] [TPS_SWCT_01425] [TPS_SWCT_01427] [TPS_SWCT_01453] [TPS_SWCT_01538] [TPS_SWCT_01540] [TPS_SWCT_01546]	[TPS_SWCT_01029] [TPS_SWCT_01132] [TPS_SWCT_01134] [TPS_SWCT_01136] [TPS_SWCT_01138] [TPS_SWCT_01140] [TPS_SWCT_01426] [TPS_SWCT_01428] [TPS_SWCT_01537] [TPS_SWCT_01539] [TPS_SWCT_01544] [TPS_SWCT_01547]

		[TPS_SWCT_01577] [TPS_SWCT_01578] [TPS_SWCT_01582] [TPS_SWCT_01591] [TPS_SWCT_01627] [TPS_SWCT_01628] [TPS_SWCT_01629] [TPS_SWCT_01630] [TPS_SWCT_01631] [TPS_SWCT_01632] [TPS_SWCT_01633] [TPS_SWCT_01634] [TPS_SWCT_01639] [TPS_SWCT_01640] [TPS_SWCT_01654] [TPS_SWCT_01655] [TPS_SWCT_01656] [TPS_SWCT_01657] [TPS_SWCT_01658] [TPS_SWCT_02002] [TPS_SWCT_02003] [TPS_SWCT_02004] [TPS_SWCT_02005] [TPS_SWCT_02007] [TPS_SWCT_02008] [TPS_SWCT_02009] [TPS_SWCT_02010] [TPS_SWCT_02011] [TPS_SWCT_02012] [TPS_SWCT_02013] [TPS_SWCT_02014] [TPS_SWCT_02015] [TPS_SWCT_02016]
[RS_SWCT_03200]	The SW-Component template shall support vehicle and application mode management	[TPS_SWCT_01008] [TPS_SWCT_01009] [TPS_SWCT_01010] [TPS_SWCT_01011] [TPS_SWCT_01063] [TPS_SWCT_01064] [TPS_SWCT_01065] [TPS_SWCT_01066] [TPS_SWCT_01067] [TPS_SWCT_01071] [TPS_SWCT_01450] [TPS_SWCT_01451] [TPS_SWCT_01552] [TPS_SWCT_01553] [TPS_SWCT_01554] [TPS_SWCT_01572] [TPS_SWCT_01581]
[RS_SWCT_03201]	The SW-Component template shall support Portgroups	[TPS_SWCT_01063] [TPS_SWCT_01064] [TPS_SWCT_01065] [TPS_SWCT_01066] [TPS_SWCT_01096] [TPS_SWCT_01126] [TPS_SWCT_01169] [TPS_SWCT_01173] [TPS_SWCT_01174]
[RS_SWCT_03202]	The SW-Component template shall support enabling SWCs to request dedicated modes	[TPS_SWCT_01086] [TPS_SWCT_01201] [TPS_SWCT_01353] [TPS_SWCT_01554] [TPS_SWCT_01572]
[RS_SWCT_03203]	The SW-Component template shall support propagation of mode information	[TPS_SWCT_01086] [TPS_SWCT_01087] [TPS_SWCT_01200] [TPS_SWCT_01201] [TPS_SWCT_01202] [TPS_SWCT_01552] [TPS_SWCT_01553] [TPS_SWCT_01566]
[RS_SWCT_03210]	The SW-Component template shall support integrity and scaling at ports	[TPS_SWCT_01023] [TPS_SWCT_01024] [TPS_SWCT_01099] [TPS_SWCT_01100] [TPS_SWCT_01101] [TPS_SWCT_01102] [TPS_SWCT_01103] [TPS_SWCT_01104] [TPS_SWCT_01105] [TPS_SWCT_01158] [TPS_SWCT_01159] [TPS_SWCT_01160] [TPS_SWCT_01161] [TPS_SWCT_01162] [TPS_SWCT_01163] [TPS_SWCT_01164] [TPS_SWCT_01165] [TPS_SWCT_01166] [TPS_SWCT_01167] [TPS_SWCT_01168] [TPS_SWCT_01449] [TPS_SWCT_01543] [TPS_SWCT_01549] [TPS_SWCT_01550] [TPS_SWCT_01551] [TPS_SWCT_01560] [TPS_SWCT_01561] [TPS_SWCT_01583]

[RS_SWCT_03215]	The SW-Component template shall define the need to add application data type on top of implementation data type	[TPS_SWCT_01072]	[TPS_SWCT_01073] [TPS_SWCT_01074] [TPS_SWCT_01189] [TPS_SWCT_01229] [TPS_SWCT_01231] [TPS_SWCT_01235] [TPS_SWCT_01236]
[RS_SWCT_03216]	The SW-Component template shall support application data type	[TPS_SWCT_01072] [TPS_SWCT_01179] [TPS_SWCT_01181] [TPS_SWCT_01184] [TPS_SWCT_01185] [TPS_SWCT_01189] [TPS_SWCT_01191] [TPS_SWCT_01229] [TPS_SWCT_01230] [TPS_SWCT_01231] [TPS_SWCT_01235] [TPS_SWCT_01236] [TPS_SWCT_01237] [TPS_SWCT_01240] [TPS_SWCT_01241] [TPS_SWCT_01242] [TPS_SWCT_01243] [TPS_SWCT_01249] [TPS_SWCT_01256] [TPS_SWCT_01486]	[TPS_SWCT_01073] [TPS_SWCT_01180] [TPS_SWCT_01183] [TPS_SWCT_01185] [TPS_SWCT_01191] [TPS_SWCT_01230] [TPS_SWCT_01235] [TPS_SWCT_01237] [TPS_SWCT_01241] [TPS_SWCT_01243] [TPS_SWCT_01256]
[RS_SWCT_03217]	The SW-Component template shall support implementation data type	[TPS_SWCT_01072] [TPS_SWCT_01183] [TPS_SWCT_01189] [TPS_SWCT_01229] [TPS_SWCT_01231] [TPS_SWCT_01232] [TPS_SWCT_01233] [TPS_SWCT_01235] [TPS_SWCT_01236] [TPS_SWCT_01237] [TPS_SWCT_01248] [TPS_SWCT_01250] [TPS_SWCT_01251] [TPS_SWCT_01252] [TPS_SWCT_01253] [TPS_SWCT_01254] [TPS_SWCT_01255] [TPS_SWCT_01257] [TPS_SWCT_01258] [TPS_SWCT_01259]	[TPS_SWCT_01074] [TPS_SWCT_01184] [TPS_SWCT_01191] [TPS_SWCT_01231] [TPS_SWCT_01233] [TPS_SWCT_01236] [TPS_SWCT_01248] [TPS_SWCT_01251] [TPS_SWCT_01253] [TPS_SWCT_01255] [TPS_SWCT_01258]
[RS_SWCT_03218]	The SW-Component template shall support data types for primitive data mapping	[TPS_SWCT_01477]	
[RS_SWCT_03220]	The SW-Component template shall allow communication attributes on compositions	[TPS_SWCT_01088]	[TPS_SWCT_01568]
[RS_SWCT_03221]	The SW-Component template shall allow port specific configuration of data transformation properties	[TPS_SWCT_01222] [TPS_SWCT_01595] [TPS_SWCT_01597] [TPS_SWCT_01599]	[TPS_SWCT_01594] [TPS_SWCT_01596] [TPS_SWCT_01598] [TPS_SWCT_01600]
[RS_SWCT_03222]	The SW-Component template shall support error notification for transformed data communication	[TPS_SWCT_01616] [TPS_SWCT_01626]	[TPS_SWCT_01624]
[RS_SWCT_03225]	The SW-Component template shall support an enhanced non-volatile (NV) memory interface	[TPS_SWCT_01141] [TPS_SWCT_01143] [TPS_SWCT_01228] [TPS_SWCT_01585] [TPS_SWCT_01587] [TPS_SWCT_01589]	[TPS_SWCT_01142] [TPS_SWCT_01227] [TPS_SWCT_01584] [TPS_SWCT_01586] [TPS_SWCT_01588] [TPS_SWCT_01590]
[RS_SWCT_03230]	The SW-Component template shall support documentation of M1 artifacts	[TPS_SWCT_01062]	

[RS_SWCT_03240]	The SW-Component template shall support end-to-end communication protection	[TPS_SWCT_01089] [TPS_SWCT_01091] [TPS_SWCT_01093] [TPS_SWCT_01095] [TPS_SWCT_01529]	[TPS_SWCT_01090] [TPS_SWCT_01092] [TPS_SWCT_01094] [TPS_SWCT_01508]
[RS_SWCT_03241]	The SW-Component template shall support partial networking	[TPS_SWCT_01169] [TPS_SWCT_01171] [TPS_SWCT_01173] [TPS_SWCT_01175]	[TPS_SWCT_01170] [TPS_SWCT_01172] [TPS_SWCT_01174]
[RS_SWCT_03250]	The SW-Component template shall support bidirectional communication	[TPS_SWCT_01112] [TPS_SWCT_01454] [TPS_SWCT_01514]	[TPS_SWCT_01113] [TPS_SWCT_01455] [TPS_SWCT_01573]
[RS_SWCT_03260]	The SW-Component template shall support rule-based initialization of arrays	[TPS_SWCT_01484] [TPS_SWCT_01493] [TPS_SWCT_01495] [TPS_SWCT_01609]	[TPS_SWCT_01485] [TPS_SWCT_01494] [TPS_SWCT_01528]
[RS_SWCT_03270]	The SW-Component template shall support overriding the activation period time on instance level	[TPS_SWCT_02507]	
[RS_SWCT_03280]	The SW-Component template shall support the description of bypass points and bypass scenarios	[TPS_SWCT_02046] [TPS_SWCT_02048] [TPS_SWCT_02050] [TPS_SWCT_02052]	[TPS_SWCT_02047] [TPS_SWCT_02049] [TPS_SWCT_02051]
[RS_SWCT_03290]	The SW-Component template shall support the initialization of runnables without usage of mode management	[TPS_SWCT_01525]	
[RS_SWCT_03310]	The SW-Component template shall support Diagnostics over IP	[TPS_SWCT_01537] [TPS_SWCT_01539] [TPS_SWCT_01546]	[TPS_SWCT_01538] [TPS_SWCT_01544]

Table 1.3: RequirementsTracing

2 Conceptual Aspects

2.1 Introduction

For the sake of a compact description of relevant meta-model elements the discussion and explanation of conceptual aspects has been concentrated in this chapter.

Reading this chapter is not a pre-requisite for understanding the subsequent chapters. It just provides a central place for the detailed description of conceptual aspects used in various other chapters of this document.

The actual explanation of the concept of a software-component starts in chapter [3](#).

2.2 Measurement and Calibration

2.2.1 Basic Approach of Measurement and Calibration

While performing the calibration process using a MCD tool (Measurement, Calibration, and Diagnostic) the calibration engineer needs to have a specific insight to the data within the CPU at runtime.

This insight is provided by access to ECU internal variables (also called measurements) as well as calibration parameters (sometimes also called characteristic value). For more details, please refer to [\[TPS_SWCT_01418\]](#)

The description of measurement variables and calibration parameters is basically the same. In AUTOSAR both appear finally as [DataPrototypes](#).

2.2.2 Calibration Parameters Overview

A Calibration Parameter is a parameter which characterizes the dynamics of a control algorithm. From a software implementation point of view, it is a variable with only read-access during the normal operation of an ECU. Characteristics are specialized [DataPrototype](#) entities in terms of its associated type but are used in a similar way.

[TPS_SWCT_01418] Ways to define a calibration parameter [This means that Calibration Parameters can be defined

- individually for a [SwComponentPrototype](#) in the [SwcInternalBehavior](#) of a [SwComponentType](#) via an aggregation of an [ParameterDataPrototype](#) in the role of [perInstanceParameter](#) (similar to [PerInstanceMemory](#)) (see chapter [2.2.3.3](#))
- sharing between all [SwComponentPrototypes](#) of the same [SwComponentType](#) in its [SwcInternalBehavior](#) via an aggregation of an [ParameterDat-](#)

aPrototype in the role of sharedParameter or constantMemory (see chapter 2.2.3.2)

- for several SwComponentPrototypes (using the port-/interface-concept with ParameterInterfaces) (see chapter 2.2.3.1).

]()

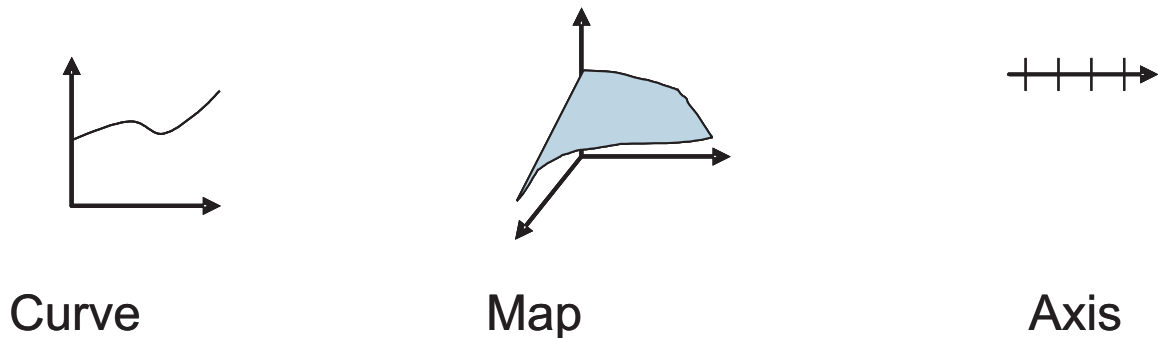


Figure 2.1: Some Categories of calibration parameters

Note: the structure of various calibration objects is visualized in [14].

2.2.3 Using Calibration Parameters

As mentioned above, a ParameterDataPrototype can be used in the context of SwcInternalBehavior as well as in the context of PortPrototypes.

2.2.3.1 Sharing Calibration Parameters within Compositions

To provide calibration parameters for being visible in other SwComponentTypes, a dedicated ParameterSwComponentType (see Figure 3.4) that inherits from SwComponentType has to be used as a SwComponentPrototype within a CompositionSwComponentType.

Class	ParameterSwComponentType			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	The ParameterSwComponentType defines parameters and characteristic values accessible via provided Ports. The provided values are the same for all connected SwComponentPrototypes Tags: atp.recommendedPackage=SwComponentTypes			
Base	ARElement, ARObjekt, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable, SwComponentType			
Attribute	Datatype	Mul.	Kind	Note

Attribute	Datatype	Mul.	Kind	Note
constantMapping	ConstantSpecificationMappingSet	*	ref	Reference to the ConstantSpecificationMapping to be applied for the particular ParameterSwComponentType Stereotypes: atpSplitable Tags: atp.Splitkey=constantMapping
dataTypeMapping	DataTypeMappingSet	*	ref	Reference to the DataTypeMapping to be applied for the particular ParameterSwComponentType Stereotypes: atpSplitable Tags: atp.Splitkey=dataTypeMapping
instantiationDataDefProps	InstantiationDataDefProps	*	aggr	The purpose of this is that within the context of a given SwComponentType some data def properties of individual instantiations can be modified. The aggregation of InstantiationDataDefProps is subject to variability with the purpose to support the conditional existence of PortPrototypes Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime

Table 2.1: ParameterSwComponentType

[TPS_SWCT_01420] **SwComponentType** requiring access to shared calibration parameters needs **RPortPrototype** typed by a **ParameterInterface** [Every **SwComponentType** requiring access to shared calibration parameters will have an **RPortPrototype** typed by a **ParameterInterface**. The definition of this shared calibration access in the context of a **CompositionSwComponentType** will be defined by creating a **SwConnector** between the relevant **SwComponentPrototypes**.]()

Class	ParameterInterface			
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
Note	A parameter interface declares a number of parameter and characteristic values to be exchanged between parameter components and software components. Tags: atp.recommendedPackage=PortInterfaces			
Base	ARElement, ARObjct, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, CollectableElement, DataInterface, Identifiable, MultilanguageReferrable, PackageableElement, PortInterface, Referrable			
Attribute	Datatype	Mul.	Kind	Note
parameter	ParameterDataPrototype	1..*	aggr	The ParameterDataPrototype of this ParameterInterface.

Table 2.2: ParameterInterface

[TPS_SWCT_01421] **ParameterInterface** is not restricted to parameters which can actually can be calibrated [Note that a **ParameterInterface** is not restricted to parameters which can actually can be calibrated. It can be used whenever there

shall be no write access to the data during normal operation of the software, i.e. only constant data are visible over the interface. $\downarrow()$

The compatibility rules for [ParameterInterfaces](#) are described in chapter 6.4; the compatibility rules for [ParameterDataPrototypes](#) are described in chapter 6.4.4.

[TPS_SWCT_01422] Delegation of [PortPrototypes](#) typed by a [Parameter-Interface](#) $\lceil$ Access to shared calibration parameters can be provided and required even over [CompositionSwComponentTypes](#) using [DelegationSwConnectors](#) and [AssemblySwConnectors](#).

This means that each access to calibration parameters between [SwComponentPrototypes](#) is explicitly visible. If a [SwConnector](#) spans after the mapping of [SwComponentPrototypes](#) over two different ECUs the system generation process has to ensure the proper allocation of the [ParameterDataPrototype](#) (see Figure 2.2) while the calibration system has to cope with setting the parameter synchronously on the affected ECUs. $\downarrow()$

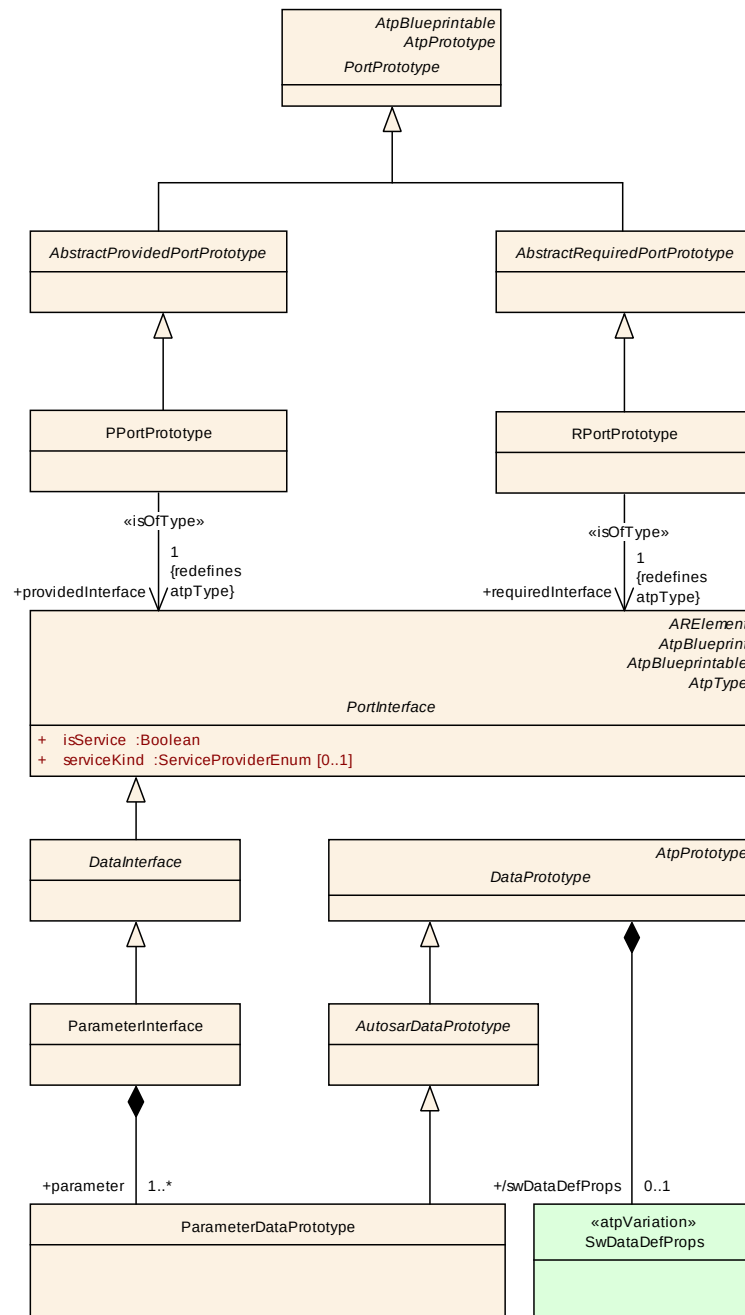


Figure 2.2: ParameterInterface

2.2.3.2 Sharing Calibration Parameters between SwComponentPrototypes of the Same SwComponentType

To share calibration parameters between several **SwComponentPrototypes** of the same **SwComponentType**, a **ParameterDataPrototype** is attached to an **SwcInternalBehavior** in **sharedParameter** role (see [TPS_SWCT_01418]).

When the `SwcInternalBehavior` is aggregated by an `AtomicSwComponentType` the actual calibration parameters of the `ParameterDataPrototype` is the same for all `SwComponentPrototypes`.

[TPS_SWCT_01423] `ParameterDataPrototype` aggregated in the role `constantMemory` [Additionally, it is possible to describe the implementation of shared characteristic values via a `ParameterDataPrototype` which is attached to an `SwcInternalBehavior` in the role `constantMemory`.

In contrast to the `ParameterDataPrototype` in `sharedParameter` role this kind of memory is not instantiated by the RTE. This supports more efficient implementations (especially for software components provided as object code) by avoidance of the additional indirection caused by the RTE's component data structure.]()

Further on this kind of memory reduces the dependencies of the software-component's implementation to generated RTE code which is appreciated for safety related functionalities.

Nevertheless the information about these characteristic values has to be taken into account for the A2L file generation.

A typical example for this kind of sharing code between instances is dealing with two lambda sensors in multiple cylinder-bank engines, where (at least) two `SwComponentPrototypes` for each lambda sensor will use the very same Calibration Parameters.

2.2.3.3 Providing Instance Individual Characteristic Data

[TPS_SWCT_01424] `ParameterDataPrototype` aggregated in the role `perInstanceParameter` [To provide instance individual calibration parameters a `ParameterDataPrototype` is owned by a `SwcInternalBehavior` in `perInstanceParameter` role.

When the `SwcInternalBehavior` is attached to an `AtomicSwComponentType`, the actual calibration values are specific for each `SwComponentPrototype`.]()

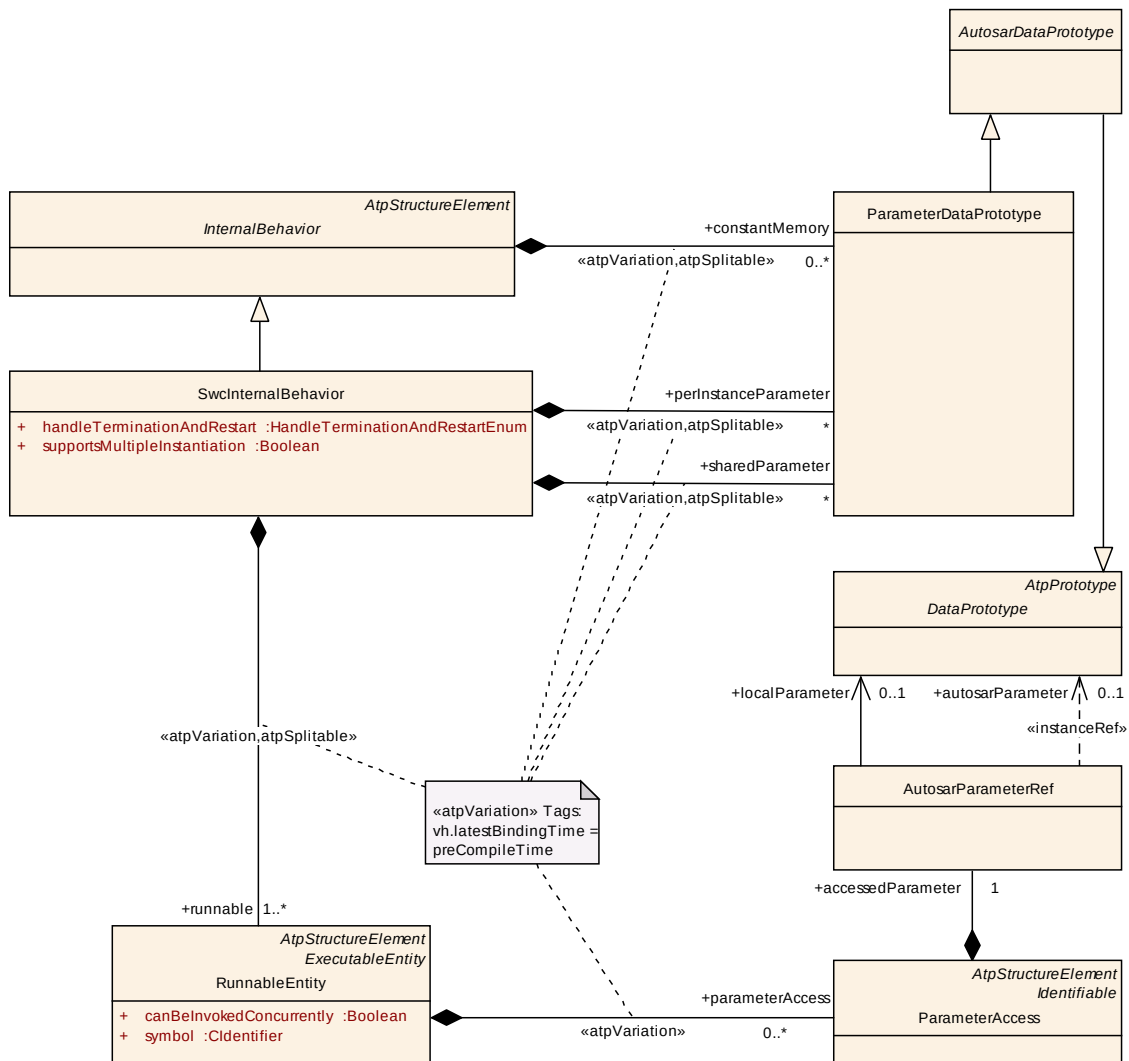


Figure 2.3: ParameterDataPrototypes in internal behavior

2.3 Runtime and Data Consistency Aspects

2.3.1 Background: the Issues

This section gives some background information and lists possible strategies concerning the implementation of the `RunnableEntity` and the RTE with respect to efficient communication between the `RunnableEntity`s.

The communication among `RunnableEntity`s can very efficiently be implemented by means of “sharing memory”¹.

¹Please note that the term “sharing memory” can be interpreted on different levels. It is e.g. in the C language possible to use variables with external linkage (a.k.a. “global variables”, although this term is not officially defined by the C language) for the purpose of inter-Runnable communication.

This is technically feasible because it is always guaranteed that the `RunnableEntities` within an `AtomicSwComponentType` are always gathered at a specific processing unit (in other words: distribution is not an option).

Note that the purpose of communication among the `RunnableEntities` is to establish a data flow scheme. The latter is a very popular pattern in the application of control theory to automotive embedded systems. So if “global variables” are used for establishing internal communication among `RunnableEntities` they acquire the semantics of so called state-messages.

Nevertheless, directly sharing memory between `RunnableEntities` requires a serious problem to be solved: the guarantee of data consistency among communicating `RunnableEntities`. The `RunnableEntities` will indeed be mapped to tasks so that one `RunnableEntity` of an `AtomicSwComponentType` may be preempted by a different `RunnableEntity` of the same `AtomicSwComponentType`.

Please note that a purist approach to achieving data consistency not only applies to single accesses of concurrently accessed variables. Rather, it would not be permitted that the value of a concurrently accessed variable (with state-message semantics) is unintentionally changed during the run-time of a `RunnableEntity`.

The following paragraphs describe some common strategies that can be used to ensure the required data-consistency. We do not attempt to describe the pros or cons of these approaches.

2.3.1.1 Mutual Exclusion with Semaphores

Multi-threaded operating systems provide mutexes (mutual exclusion semaphores) that protect access to an exclusive resource that is used from within several tasks.

The RTE could use these OS-provided mutexes to make sure that the `RunnableEntities` sharing a memory-space would never run concurrently. The RTE would make sure the task running the `RunnableEntity` has taken an appropriate mutex before accessing the memory shared between the `RunnableEntities`.

2.3.1.2 Interrupt Disabling

Another alternative would be the disabling of interrupts during the run-time of `RunnableEntities` or at least for a period in time identical to the interval from the first to the last usage of a concurrently accessed variable in a `RunnableEntity`. This approach could lead to seriously non-deterministic execution timing.

2.3.1.3 Priority Ceiling

Priority ceiling allows for a non-blocking protection of shared resources. Provided that the priority scheme is static, the AUTOSAR OS is capable of temporarily raising the priority of a task that attempts to access a shared resource to the highest priority of all tasks that would ever attempt to access the resource.

By this means is technically impossible that a task in temporary possession of a resource is ever preempted by a task that attempts to access the resource as well.

2.3.1.4 Implicit Communication by Means of Variable Copies

Another alternative is the usage of copies of concurrently accessed variables with state message semantics. Note that this approach directly corresponds to the semantics of “implicit” sender-receiver communication (see 7.5.1.2).

This means in particular that for a concurrently used variable a copy is created on which a `RunnableEntity` entity can work without any danger of data inconsistency.

This concept requires additional code to write the value of the concurrently accessed variable to the copy before the `RunnableEntity` that accesses the variable is executed. The value of the copy shall be written back to the concurrently accessed variable after the `RunnableEntity` has been terminated.

This concept is sketched in Figure 2.4. Since it would be too expensive and error-prone to manually care about the copy routines it would be a good idea to leave the creation of the additional code to a suitable code generator.

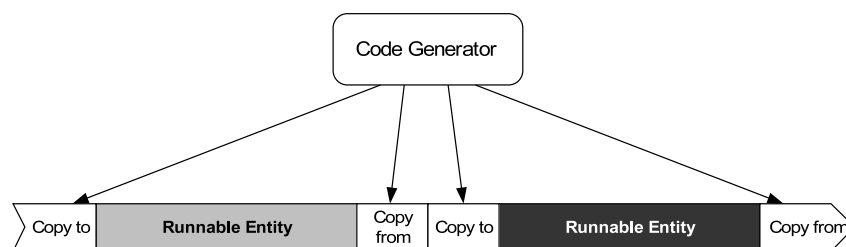


Figure 2.4: Generation of copy routines around `RunnableEntity`s

The additional copy routines as sketched in Figure 2.4 already protect the particular `RunnableEntity`s from unintended changes of concurrently accessed variables. It would, however, be possible to further optimize the process by reducing the additional code at the beginning and end of each task (see Figure 2.5).

2.3.2 Data Consistency at Runtime

In addition, copy routines will only be inserted where appropriate, e.g. a copy routine for writing the value of a copy back to the concurrently accessed variable will only be inserted if the `RunnableEntity` has write access to the concurrently used variable.

Please note that the copy routines have to temporarily make sure that the copy process is not interrupted in order to be capable of consistently copying the values from and to the concurrently accessed variable.

These periods, however, are supposed to be very short compared with the overall run-time consumption of the `RunnableEntity` and thus would not have a significant impact on the runtime behavior.

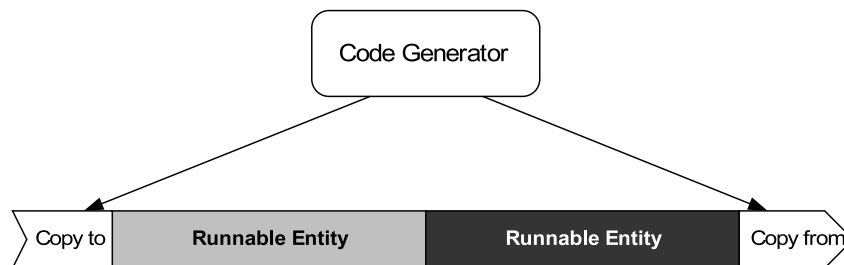


Figure 2.5: Optimized insertion of copy routines

Further optimization criteria can be applied, for example: it would be perfectly safe to avoid the creation of copies for `RunnableEntity`s that are scheduled in the task with the highest priority of all tasks that (via contained `RunnableEntity`s) access a certain concurrently accessed variable.

In order to keep the application code free of any dependencies from the code generation, access to concurrently accessed variables will be guarded by macros that are later resolved by the code generator.

The presence of the guard macros directly supports the reuse on the level of source code. The reuse on the level of object code is only possible if the scheduling scenario (in terms of the assignment of `RunnableEntity`s to priority levels) does not change.

This concept can only be implemented properly with the aid of a code generator if the variables in question can be identified. In other words: the description of an `Atomic-SwComponentType` has to expose all concurrently accessed variables to the outside world.

2.3.3 Modeling Aspects of Data Consistency

The intrinsic meaning of the terms “explicit communication” and “implicit communication” is explained in section 7.5.1.1. It would be fair to say that the distinction between implicit and explicit communication establishes a usage pattern in the application do-

main, i.e. in the world of the developer of AUTOSAR software-components and their implementation.

There is another facet to this subject, however, namely the question how this pattern is implemented in the meta-model. With respect to the application of the pattern for port-based communication the details can be found in section 7.5.1.2, more specifically in section 7.5.1.3. The consideration of the internal communication based on so-called “inter-runnable variables” is described in section 7.4.2.

By reading the respective text sections it becomes apparent that the two applications of the pattern are modeled differently. The port-based communication uses the `VariableAccess` to formalize different roles of accessing communication elements. Some of the roles used for this purpose imply explicit communication (e.g. `dataSendPoint`) and some represent implicit communication (e.g. `dataWriteAccess`).

The important thing about using the `VariableAccess`, however, is that the modeling of communication roles is abstracted from the actual communication elements and represents a uniform (meaning: it can refer to the target directly or by a so-called `InstanceRef`) modeling approach that is applied for all use cases².

Admittedly, this is handled in a different way for the internal communication. Here, the additional layer of abstraction is not used (although it would have been technically feasible to do so) with respect to the clear separation of “inter-runnable variables with implicit behavior” and “inter-runnable variables with explicit behavior” in the RTE. The implementation of different communication roles (i.e. implicit vs. explicit) is done by directly aggregating `VariableDataPrototype` in the roles `explicitInterRunnableVariable` and `implicitInterRunnableVariable`.

On the other hand, access to internal communication **never** requires the usage of an `InstanceRef` and therefore the abstraction might be considered unnecessary overhead that blows up the M1 model.

2.4 Variant Handling in the Software Component Template

The Software Component Template supports the creation of *Variants* in a subset of its model elements. The full list of model elements that support variation can be found in the appendix.

[TPS_SWCT_01038] Support for Variant Handling in the in Software Component Template [The Variant Handling support in the in Software Component Template is mainly driven by the purpose to describe a variable system on Virtual Functional Bus[3] level by varying

- the existence of `SwComponentPrototypes`
- the existence of `SwConnectors`

²On a related note, even for non-communication related data access the same pattern applies implemented by `ParameterAccess`

- the existence of [Chapters](#) of [SwComponentDocumentation](#)
- the existence of [PortPrototypes](#)

]([RS_SWCT_00220](#), [RS_SWCT_03100](#), [RS_SWCT_03140](#), [RS_SWCT_03142](#), [RS_SWCT_03154](#))

[TPS_SWCT_01039] Purpose of variant handling [This supports adjusting the number and kind of software-component instances as well as their interconnection in a particular system variant.]([RS_SWCT_00220](#))

[TPS_SWCT_01447] Applicable binding times for model elements in the scope of the Software Component Template [The first three cases are supporting *PostBuild* binding. For the existence of [PortPrototypes](#) only [preCompileTime](#) is supported as latest [Binding Time](#).]([RS_SWCT_00220](#))

[TPS_SWCT_01040] [SwConnector](#) exists depending on a *PostBuild* condition [A [SwConnector](#) which exists depending on a *PostBuild* condition has an impact on the behavior of API function calls that apply on a [PortPrototype](#) to which the [SwConnector](#) is attached. If the [SwConnector](#) does not exist the behavior of the RTE API functions need to take this into account. This means that the RTE implementation of this [PortPrototype](#) resembles the behavior of an unconnected [PortPrototype](#).]([RS_SWCT_00220](#), [RS_SWCT_03100](#), [RS_SWCT_03143](#))

Please find more details in the specification of the RTE [2].

[TPS_SWCT_01041] API functions of not existing [SwConnector](#) are still part of the software-component's implementation [If [SwConnectors](#) do not exist the corresponding API functions are still part of the software-component's implementation. It is not possible to remove the API functions in a *PostBuild* step. Therefore the latest reasonable [Binding Time](#) for the conditional existence of a [PortPrototype](#) is [preCompileTime](#).]([RS_SWCT_00220](#), [RS_SWCT_03100](#))

[TPS_SWCT_01085] Variation on the behavior level [In addition to variation of the VFB-related model elements, the description of variant software-component implementations is supported. Please note that this requires a broad support of variability in the *Internal Behavior*.

The identified main use case are

- the existence of [RunnableEntitys](#)
- the existence of [RTEEvents](#)
- the existence of [VariableDataPrototypes](#) in the roles [implicitInterRunnableVariable](#) and [explicitInterRunnableVariable](#)
- the existence of [ParameterDataPrototypes](#) in the roles [perInstanceParameter](#), [sharedParameter](#), and [constantMemory](#)

]([RS_SWCT_03149](#), [RS_SWCT_03150](#), [RS_SWCT_03151](#), [RS_SWCT_03153](#))

For the same reason that applies on the existence of [PortPrototype](#) the latest Binding Time of these kinds of variability is [preCompileTime](#).

In the meta-model, all locations that may exhibit variability are marked with the stereotype `<<atpVariation>>`. This allows the definition of possible variation points. Tagged Values are used to specify additional information, for example the latest binding time.

[TPS_SWCT_01042] Four types of locations in the meta-model which may exhibit variability [There are four types of locations in the meta-model which may exhibit variability:

- Aggregations
- Associations
- Attribute Values
- Classes providing property sets

]([RS_SWCT_00220](#), [RS_SWCT_03100](#))

The reasons for the attachment of the stereotype `<<atpVariation>>` to certain model elements and the consequences for other model elements are explained in class tables in the following chapters. More details about the AUTOSAR Variant Handling Concept can be found in the AUTOSAR Generic Structure Template [12].

2.5 Communication Specification of Composition Component Types

[TPS_SWCT_01088] ComSpecs defined by [CompositionSwComponentTypes](#) [It shall be possible to attach `ComSpecs` to [PortPrototypes](#) owned by [CompositionSwComponentTypes](#).]([RS_SWCT_03220](#))

2.5.1 Rationale

`ComSpecs` attached to a [PortPrototype](#) owned by an [AtomicSwComponentType](#) have a direct impact on the generation of the RTE. The RTE Generator, on the other hand, does not consider the existence of [CompositionSwComponentTypes](#).

Nevertheless, there are some cases where the definition of a `ComSpec` attached to a [PortPrototype](#) owned by a [CompositionSwComponentType](#) does make sense.

That is, in case an OEM wants to submit the definition of a [CompositionSwComponentType](#) to a supplier for adding more details and implementing the behavior the OEM might want to point out that from the OEM's point of view sender `initValues` and receiver `initValues` apply for the elements of [PortInterfaces](#) used to type the delegation [PortPrototypes](#).

The idea is that the supplier takes over the `initValues` attached to the delegation `PortPrototypes` and *copies* them to the `PortPrototypes` owned by `SwComponentPrototypes` of the `CompositionSwComponentType`.

[TPS_SWCT_01568] Consideration of `RPortComSpec` or `PPortComSpec` depending on the ownership [The RTE Generator shall take the attributes of the `RPortComSpec` or `PPortComSpec` of the `PortPrototypes` owned by `AtomicSwComponentTypes` or `ParameterSwComponentType` and ignore the attributes of the `RPortComSpec` or `PPortComSpec` attached to `PortPrototypes` owed by `CompositionSwComponentType`.] (*RS_SWCT_03220*)

Therefore, the `initValues` of the delegation `PortPrototype` would be taken as *mere templates* for the detailing of `PortPrototypes` connected to the delegation `PortPrototypes`.

It is not required that the `initValues` of delegated `PortPrototype` and a `PortPrototype` connected by means of a `DelegationSwConnector` match.

Although this would certainly make sense in many cases it is eventually still left to the supplier to decide on the specific `initValues` applicable inside the `CompositionSwComponentType`.

On the other hand, a requirement that the `initValues` defined on the surface of `CompositionSwComponentType` and the inside of the `CompositionSwComponentType` shall be consistent in any case might effectively prevent the reuse of existing `AtomicSwComponentTypes`.

Please note that the ability to define a `ComSpec` in the context of a `CompositionSwComponentType` implies that it shall be possible to define mappings of `ApplicationDataTypes` used in a `PortInterface` to their corresponding `ImplementationDataTypes`.

For this purpose the `CompositionSwComponentType` owns a `DataTypeMappingSet` in the role `dataTypeMapping` and a `ConstantSpecificationMappingSet` in the role `constantValueMapping`.

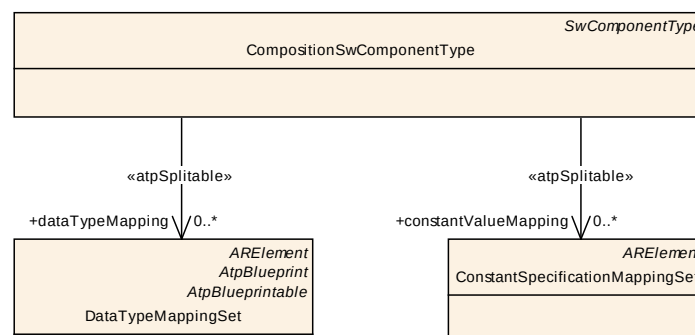


Figure 2.6: Specification of data type mapping for `CompositionSwComponentType`

2.6 PRPortPrototype

In some cases [SwComponentType](#)s need to read and write the same piece of data. One of the most prominent examples for this use case is the [NvBlockSwComponentType](#) that factually reads and writes blocks of NvRAM.

Without the ability to combine read and write semantics in a kind of [PortPrototype](#) that supports both read and write semantics work-arounds have to be implemented that come with a certain footprint on memory and processing time.

2.6.1 Use Case 1

Without the ability to define a combined read and write semantics the definition of an [RPortPrototype](#) and a [PPortPrototype](#) is required for reading and writing the applicable data.

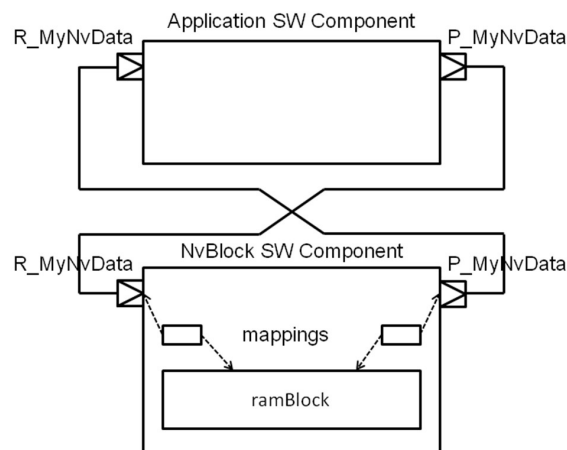


Figure 2.7: Use Case 1 for the existence of [PRPortPrototype](#)

Technically, this read and write access is related to the same data item in an NVRAM Block. This requires a consistent connection of the [PortPrototypes](#) between an [NvBlockSwComponentType](#) and [ApplicationSwComponentType](#) as well as a consistent mapping of the corresponding [RPortPrototype](#) and a [PPortPrototype](#) of the [NvBlockSwComponentType](#) and the related element of the [ramBlock](#).

2.6.2 Use Case 2

It may happen that a [SwComponentType](#) need to consume the same data that it produces. If the only way to achieve this was the connection of a [PPortPrototype](#) to an [RPortPrototype](#) of the same [SwComponentType](#) then the creator of the [SwComponentType](#) cannot enforce this connection as it is created on a higher level of abstraction in the context of a [CompositionSwComponentType](#).

In other words, it is impossible to fully specify the semantics of the otherwise self-contained `SwComponentType`.

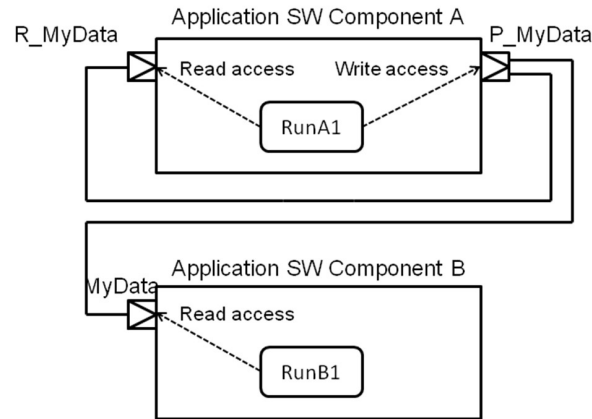


Figure 2.8: Use Case 2 for the existence of `PRPortPrototype`

This means that only in the in best case one buffer for the data is needed. But depending on the mapping `RunnableEntitys` to OS tasks additional buffers may need to be allocated by the RTE to fully implement the implicit communication pattern.

As an alternative, the `ApplicationSwComponentType` could utilize inter-runnable variables but unfortunately this inhibits any optimization in the RTE and will consume additional RAM. In contrast to the previous approach at least two buffers are needed.

2.6.3 Use Case 3

In this scenario, several `ApplicationSwComponentTypes` are iterating over the same large set of data. This means each `ApplicationSwComponentType` implements one out of many steps of a complex data processing algorithm applied to the same piece of data.

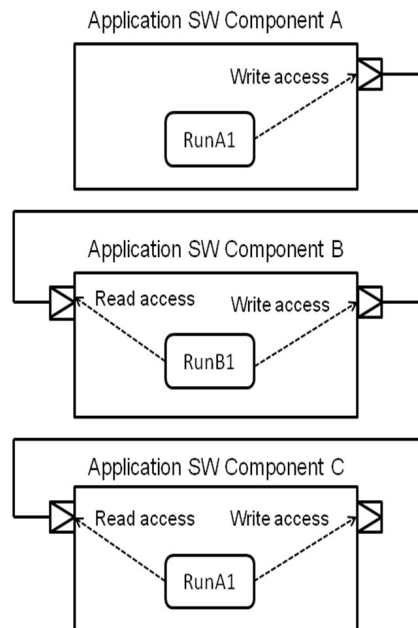


Figure 2.9: Use Case 3 for the existence of [PRPortPrototype](#)

For example, this scenario may apply for video signal processing in camera applications. Typically, such applications will **not** be distributed over several ECUs.

It is clear that in this case the allocation of several buffers in the RTE is required to implement the individual connections between the [ApplicationSwComponentTypes](#). In most cases, the processing has to be executed at a certain point in time in a dedicated order.

2.6.4 Solution

The solution to the above-mentioned use cases is the ability to define a [PortPrototype](#) that can read and write the same piece of data. This solves both the described problem of resource consumption as well as the problem of having to define multiple [PortPrototypes](#) as outlets for same piece of data item.

The technical details of the definition of [PRPortPrototype](#) are explained in chapters [3.1](#) and [4.1](#).

2.7 Pretended Networking

[TPS_SWCT_01510] The role of pretended networking [Pretended networking is a feature to reduce energy consumption of an ECU by switching the ECU in a mode called [Pretended Networking](#). In this mode the communication on communication networks is reduced and the ECU can go into power saving modes.

When communication via communication networks is required the mode `Pretended Networking` shall be left by request of a mode change to `Normal Mode`. `()`

[TPS_SWCT_01511] Configuration option is encoded into `ModeDeclaration` [The identification of different configuration options for `Pretended Networking` shall be encoded into the definition of dedicated `ModeDeclarations` inside a `ModeDeclarationGroup`.]([RS_SWCT_03110](#))

For example, assume that an implementation of pretended networking supports three configuration options:

- `PRETENDED_NW_MODE_OFF`
- `PRETENDED_NW_MODE_ONE`
- `PRETENDED_NW_MODE_TWO`

In this example case, a `ModeDeclarationGroup` consisting of three `ModeDeclarations` shall be defined where each `ModeDeclaration` shall represent one of the above-mentioned configuration options. The `shortNames` of the `ModeDeclaration` shall be taken from the above-mentioned list.

[TPS_SWCT_01512] Request change of `Pretended Networking` mode [A `SwComponentType` that needs to be able to request a change in the operating mode of `Pretended Networking` shall provide a `PPortPrototype` typed by a `SenderReceiverInterface` (see [\[TPS_SWCT_01086\]](#)) for requesting a change (towards the `BswM` [15]) of the `Pretended Networking` mode.

It is out of the scope of this document to define the particular properties of the applicable `SenderReceiverInterface`. The details of this specific `SenderReceiverInterface` can be found in the specification of the `BswM` [15].]([RS_SWCT_03110](#))

More details about how a mode change is requested can be found in section 9.

[TPS_SWCT_01513] React on the change of `Pretended Networking` mode [A `SwComponentType` that needs to be able to react on a change in the operating mode of `Pretended Networking` shall provide an `RPortPrototype` typed by a `ModeSwitchInterface` (see [\[TPS_SWCT_01087\]](#)) for reacting on a change (initiated by the `BswM` [15]) of the `Pretended Networking` mode.

It is out of the scope of this document to define the particular properties of the applicable `ModeSwitchInterface`. The details of this specific `ModeSwitchInterface` can be found in the specification of the `BswM` [15].]([RS_SWCT_03110](#))

2.8 Variable-size Array Data Types

2.8.1 Overview and Use cases

AUTOSAR supports the definition of array data types where the size of the actual payload varies at run-time. As far as the configuration is concerned, it is possible to specify a maximum number of array elements that shall not be exceeded at run-time.

In order to properly understand the approach, it is necessary to understand that the support for `Variable-Size Array Data Types` has been introduced in two waves that each had a different motivation.

2.8.1.1 “Old-world” dynamic-size Arrays

In the first wave, the support for `Variable-Size Array Data Types` was limited to data types that basically boil down to an array where the base type is an unsigned integer data type with a length of exactly one byte.

The main use cases for this scenario are derived from diagnostics requirements as well as support for the J1939 communication protocol.

In both cases the actual length of a `Variable-Size Array Data Type` could be determined from the context, i.e. either by the diagnostic basic-software module or by the implementation of the J1939 TP.

For the lack of a better terminology, this specification distinguishes between “old-world” dynamic-size arrays and “new-world” `Variable-Size Array Data Types`. It will be necessary to clearly define the characteristics that allow for an disambiguation between the “old-world” dynamic-size arrays and “new-world” `Variable-Size Array Data Types`.

[TPS_SWCT_01641] Definition of an “old-world” dynamic-size array data type by means of an `ApplicationArrayType` [An `ApplicationArrayType` that **doesn't define** attribute `dynamicArraySizeProfile` and that aggregates an `ApplicationArrayElement` where attribute `arraySizeSemantics` exists and is set to the value `variableSize` shall be considered an “old-world” dynamic-size array data type.] (*RS_SWCT_03181*)

Please note that [TPS_SWCT_01641] can't go any deeper into the specifics of the given data type because it is intentionally focused on `ApplicationDataTypes`. There are use cases where the distinction between “old-world” dynamic-size arrays and “new-world” `Variable-Size Array Data Types` must be done in the absence of a corresponding `ImplementationDataType`.

In general, the disambiguation becomes multi-faceted (but not necessarily easier) if the definition of a corresponding `ImplementationDataType` is available (see [TPS_SWCT_01642]).

[TPS_SWCT_01642] Definition of an “old-world” dynamic-size array data type by means of an `ImplementationDataType` [An `ImplementationDataType` that (after all type references are resolved) fulfills all of the following conditions shall be considered an “old-world” dynamic-size array data type:

- The value of attribute `category` is set to `ARRAY`
- The `ImplementationDataType` doesn't define the attribute `dynamicArray-SizeProfile`
- The `ImplementationDataType` aggregates a `subElement` where
 - attribute `arraySizeSemantics` exists and is set to the value `variable-Size`
 - attribute `arraySizeHandling` does not exist
- The `ImplementationDataType.swDataDefProps.baseType` exists and the attribute
 - `baseTypeEncoding` exists and is set to the value `NONE`
 - `baseTypeSize` exists and is set to the value `8`

](*RS_SWCT_03181*)

By and large, the defining characteristics for “old-world” dynamic-size arrays is the **absence** of a definition of the attribute `ApplicationArrayDataType.dynamicArraySizeProfile` resp. `ImplementationDataType.dynamicArraySizeProfile`.

By regulation of [constr_1387], “old-world” dynamic-size arrays are not supported for transmission by means of a data transformer. The only supported kind of `Variable-Size Array Data Type` that can be transmitted using a data transformer is the “new-world” variable-size arrays.

2.8.1.2 “New-world” variable-size Arrays

In contrast to this, the second wave of support for `Variable-Size Array Data Types` was motivated by the application software layer itself.

Here, the situation is entirely different because the actual size cannot be determined by any context software module. The application itself is responsible for maintaining the proper length of a `Variable-Size Array Data Type` at run-time.

As a consequence, the specification of the actual array size at run-time needs to be reflected by the structure of the data types used for hosting the `Variable-Size Array Data Type`.

[TPS_SWCT_01644] Definition of a “new-world” variable-size array data type by means of an `ApplicationArrayDataType` [An `ApplicationArrayDataType`

that fulfills all of the following conditions shall be considered an “new-world” dynamic-size array data type.

- The `ApplicationArrayDataType` **defines** attribute `ApplicationArrayDataType.dynamicArraySizeProfile`.
- `ApplicationArrayDataType` aggregates an `ApplicationArrayElement` that **defines** attribute `ApplicationArrayElement.arraySizeHandling`.

]([RS_SWCT_03181](#))

[TPS_SWCT_01645] Definition of a “new-world” variable-size array data type by means of an `ImplementationDataType` [An `ImplementationDataType` that fulfills all of the following conditions shall be considered an “new-world” dynamic-size array data type.

- The `ImplementationDataType` **defines** attribute `ImplementationDataType.dynamicArraySizeProfile`.
- `ImplementationDataType` aggregates an `ImplementationDataTypeElement` that **defines** attribute `ImplementationDataTypeElement.arraySizeHandling`.

]([RS_SWCT_03181](#))

In contrast to the first use case described above, the application-motivated `Variable-Size Array Data Type` cannot be limited in terms of the base type of the array data type, i.e. limiting the underlying data type to an unsigned integer data type with a length of exactly one byte is not an option.

On top of that, several possible structures of `Variable-Size Array Data Types` have been required. This aspect is depicted in Figure 2.10.

[TPS_SWCT_01636] Definition of profiles for the definition of `Variable-Size Array Data Types` [The possible variants for `Variable-Size Array Data Types` are:

Linear The data type of the elements of the `Variable-Size Array Data Type` itself does not consist of a `Variable-Size Array Data Type`. This case corresponds to the tag (a) in Figure 2.10.

This case corresponds to the possible value **VSA_LINEAR** of attribute `dynamicArraySizeProfile`.

Square The data type of the elements of the `Variable-Size Array Data Type` itself consists of `Variable-Size Array Data Types` where the maximum number of elements in all “second order” arrays is **identical** to the maximum number of elements in the “first order” array. This case corresponds to the tag (b) in Figure 2.10.

This case corresponds to the possible value **VSA_SQUARE** of attribute `dynamicArraySizeProfile`.

Rectangular The data type of the elements of the [Variable-Size Array Data Type](#) itself consists of [Variable-Size Array Data Types](#) where the maximum number of elements in “second order” arrays is **identical** but this value is typically **not identical**³ to the maximum number of elements in the “first order” array. This case corresponds to the tag (c) in Figure 2.10.

This case corresponds to the possible value **VSA_RECTANGULAR** of attribute [dynamicArraySizeProfile](#).

Fully Flexible The data type of the elements of the [Variable-Size Array Data Type](#) itself consists of [Variable-Size Array Data Types](#) where the maximum number of elements in “second order” arrays is **not necessarily identical** with each other and (obviously) **not necessarily identical** to the maximum number of elements in the “first order” array. This case corresponds to the tag (d) in Figure 2.10.

This case corresponds to the possible value **VSA_FULLY_FLEXIBLE** of attribute [dynamicArraySizeProfile](#).

]([RS_SWCT_03181](#))

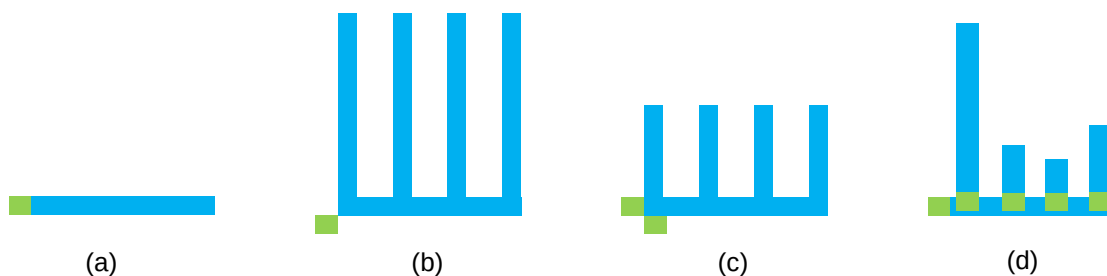


Figure 2.10: Structural variety of array data types with variable size

Please note that the leaf elements in a [Variable-Size Array Data Type](#) doesn't have to be primitive data types. As mentioned before, it is possible to define multiple-dimension [Variable-Size Array Data Types](#).

The “terminal” elements can be recognized as such in that they don't establish further [Variable-Size Array Data Types](#).

2.8.2 Modeling Aspects regarding Application Data Types

In the context of the AUTOSAR layered data type concept, the level of [Application-DataTypes](#) is not concerned about the structure of how the [Variable-Size Array Data Types](#).

³If it was, the case boils down to the rectangular scenario tagged (b).

In other words, aspects of the implementation of this kind of data type is intentionally abstracted as much as possible in order to support the idea behind the definition of `ApplicationDataTypes` as a concept that is independent from an implementation to the applicable degree.

Consequently, the support for `Variable-Size Array Data Types` on the level of `ApplicationDataTypes` requires the addition of a couple of additional attributes. Details can be found in chapter 5.2.4.2.

If a `Variable-Size Array Data Type` is modeled on the level of `Application-DataType` it is necessary to also provide a companion `ImplementationDataType` as well as a `DataTypeMap` that refers to both the `ApplicationDataType` and the `ImplementationDataType`.

The contrary is **not applicable**, i.e. it is possible to define a `Variable-Size Array Data Type` with only an `ImplementationDataType`, see [TPS_SWCT_01622].

2.8.3 Modeling Aspects regarding Implementation Data Types

On the other hand, the data type used for the actual hosting of the `Variable-Size Array Data Type` corresponds directly to the level of the `Implementation-DataType`.

Here, it is possible to define how an `ImplementationDataType` can be used to define a `Variable-Size Array Data Type`.

The definition of `ImplementationDataType` in the AUTOSAR meta-model comes with a certain level of generic nature the support for `Variable-Size Array Data Types` on this level comes as a mixture of dedicated attributes in the meta-model and a set of recipes how to support different use cases of `Variable-Size Array Data Types`.

This means that the definition of `ImplementationDataTypes` for the purpose of creating `Variable-Size Array Data Types` only has a chance to take off if the structure of these data types is replicated in different implementations of AUTOSAR software.

Therefore, AUTOSAR defines a common way of how `ImplementationDataTypes` for the purpose of creating `Variable-Size Array Data Types` shall be defined such that the `ImplementationDataType` shall be of `category` `STRUCTURE` with the following sub-elements:

1. A numerical value that determines the actual size. This element shall be called the `Size Indicator` throughout this document.
2. An array of the base-type of the `Variable-Size Array Data Type` that implements the payload of the `Variable-Size Array Data Type`. The dimension of the array shall be defined such that the intended maximum number of elements fits in.

A `Size Indicator` of a `Variable-Size Array Data Type` holds the number of valid elements of the array. This information is necessary for the RTE to handle the array efficiently.

On the sender-side this indicator is actively updated by the software-component which is the only instance that knows how many elements of the array are valid.

So the number of valid elements and the `Size Indicator` have to be kept consistent by the application. When the software-component sends the data over the RTE the RTE hands the data over to the transformer.

The transformer may evaluate the `Size Indicator` (depends on the transformer) and only work on the valid array elements. The output of the transformer can vary in length and only contain necessary data. Therefore it can be more resource saving.

On the receiver side, the last transformer in the execution order restores the data elements of the array and the value of the `Size Indicator`. This output is handed over by the RTE to the software-component. The application now is aware of the number of valid elements in the array.

The details of how `ImplementationDataTypes` need to be modeled for the implementation of `Variable-Size Array Data Types` can be found in chapter 5.2.5 and a couple of examples is available in the appendix E.1.

3 Overview: Software Components, Ports, and Interfaces

3.1 Introduction

The detailed introduction of all aspects of the `Software Component Template` in one move is considered too complex. This chapter therefore provides an overview of the main conceptual aspects of software components, ports and interfaces. The overview will then be broken down into further details in chapter 4.

One of the goals of the AUTOSAR concept is the support of re-usability on the level of application software. In other words: it should be possible to re-use existing artifacts to create further model elements instead of being forced to create every single modeling detail from scratch. One of the consequences of this approach is the application of the so-called type-prototype pattern [12].

Among other things, this concept allows for creating hierarchical structures of software-components with arbitrary complexity. However, the creation of hierarchical structures itself does not have an impact on the run-time behavior of the overall system. The actual behavior is completely defined within the individual software-components.

This conclusion is backed by the understanding that software-components are developed against the so-called *Virtual Functional Bus* (VFB), an abstract communication channel without direct dependency on ECUs and communication buses. The VFB does not provide any means for expressing a hierarchy of software-components.

Of course, the usage of the VFB has further consequences on the design of software-components which shall not directly call the operating system or the communication hardware. As a result, software-components can be deployed to actual ECUs at a rather late stage in the development process.

In order to make the description more precise, the following text preferably uses accurate meta-model terms instead of the rather vague terminology of “composition” and “software-component”.

3.2 Software Component

3.2.1 Overview

Application software within AUTOSAR is organized in self-contained units called `AtomicSwComponentTypes`. Such `AtomicSwComponentTypes` encapsulate the implementation of their functionality and behavior and merely expose well-defined connection points, called `PortPrototypes`, to the outside world.

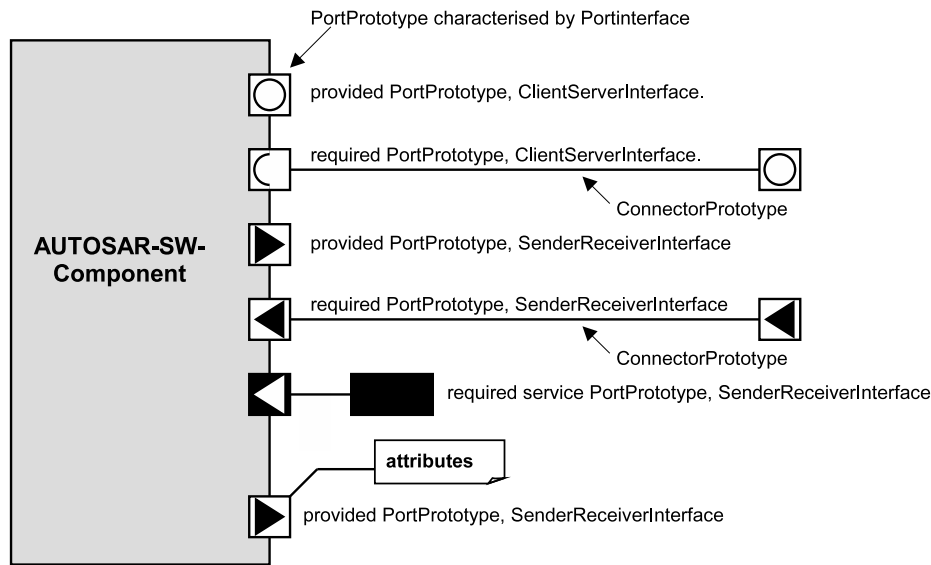


Figure 3.1: Graphical representation of software-components in AUTOSAR

The graphical appearance of AUTOSAR software-components according to [3] is depicted in Figure 3.1.

Class	SwComponentType (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	Base class for AUTOSAR software components.			
Base	ARElement, ARObjekt, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Attribute	Datatype	Mul.	Kind	Note
consistencyNeeds	ConsistencyNeeds	*	aggr	This represents the collection of ConsistencyNeeds owned by the enclosing SwComponentType. Stereotypes: atpSplittable; atpVariation Tags: atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime
port	PortPrototype	*	aggr	The ports through which this component can communicate. The aggregation of PortPrototype is subject to variability with the purpose to support the conditional existence of PortPrototypes. Stereotypes: atpSplittable; atpVariation Tags: atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime
portGroup	PortGroup	*	aggr	A port group being part of this component. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime

Attribute	Datatype	Mul.	Kind	Note
swComponentDocumentation	SwComponentDocumentation	0..1	aggr	This adds a documentation to the SwComponentType. Stereotypes: atpSplittable; atpVariation Tags: atp.Splitkey=swComponentDocumentation, variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=-10
unitGroup	UnitGroup	*	ref	This allows for the specification of which UnitGroups are relevant in the context of referencing SwComponentType.

Table 3.1: SwComponentType

3.2.2 PortPrototype

Please note that [PortPrototypes](#) of a [SwComponentType](#) are supposed to be used for attaching [SwConnectors](#) that establish an actual connection between [SwComponentPrototypes](#) (see chapter 3.3).

[TPS_SWCT_01002] [SwComponentTypes](#) may only interact by means of their [PortPrototypes](#) [[AtomicSwComponentTypes](#) (and also the more general [SwComponentTypes](#) may only interact by means of their [PortPrototypes](#)). Hidden communication dependencies that are *not* expressed by means of [PortPrototypes](#) are strictly forbidden.] ([RS_SWCT_00020](#), [RS_SWCT_00030](#), [RS_SWCT_00150](#), [RS_SWCT_00160](#), [RS_SWCT_00200](#), [RS_SWCT_00210](#), [RS_SWCT_02010](#), [RS_SWCT_02030](#))

Therefore, software-components are in theory exchangeable as long as they implement the same functionality and provide the same public communication interface to the remaining system.

Class	PortPrototype (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	Base class for the ports of an AUTOSAR software component. The aggregation of PortPrototypes is subject to variability with the purpose to support the conditional existence of ports.			
Base	ARObject, AtpBlueprintable, AtpFeature, AtpPrototype, Identifiable , Multilanguage Referrable, Referrable			
Attribute	Datatype	Mul.	Kind	Note
clientServerAnnotation	ClientServerAnnotation	*	aggr	Annotation of this PortPrototype with respect to client/server communication.
delegatedPortAnnotation	DelegatedPortAnnotation	0..1	aggr	Annotations on this delegated port.

Attribute	Datatype	Mul.	Kind	Note
ioHwAbstractionServerAnnotation	IoHwAbstractionServerAnnotation	*	aggr	Annotations on this IO Hardware Abstraction port.
modePortAnnotation	ModePortAnnotation	*	aggr	Annotations on this mode port.
nvDataPortAnnotation	NvDataPortAnnotation	*	aggr	Annotations on this non volatile data port.
parameterPortAnnotation	ParameterPortAnnotation	*	aggr	Annotations on this parameter port.
senderReceiverAnnotation	SenderReceiverAnnotation	*	aggr	Collection of annotations of this ports sender/receiver communication.
triggerPortAnnotation	TriggerPortAnnotation	*	aggr	Annotations on this trigger port.

Table 3.2: PortPrototype

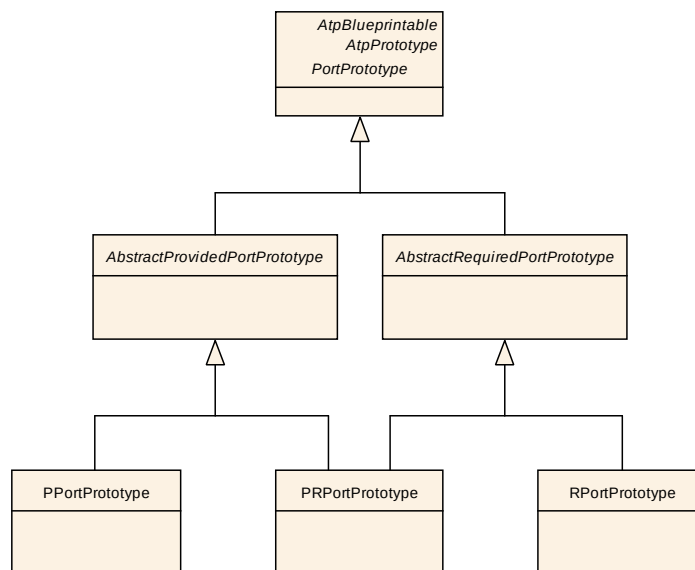


Figure 3.2: Overview of PortPrototype

[TPS_SWCT_01111] **PortPrototypes** need an additional model artifact, the **PortInterface** [Please note that **PortPrototypes** actually need an additional model artifact, the **PortInterface**, for fully describing the details of the **PortPrototype**. The concept of the **PortInterface** as another means for establishing a high degree of re-usability is described in chapter 3.4.] ([RS_SWCT_00010](#))

[TPS_SWCT_01112] **Semantics of PortPrototypes** [As depicted in Figure 3.2, **PortPrototypes** can have the following semantics:

- A require-port (in technical terms: **RPortPrototype**) requires certain services or data.

- A provide-port (or [PPortPrototype](#)) on the other hand provides services or data.
- A provide-require-port (or [PRPortPrototype](#)) combines the ability to provide and require services or data in one entity.

]([RS_SWCT_03250](#))

[TPS_SWCT_01573] A [PRPortPrototype](#) is never considered unconnected
 [A [PRPortPrototype](#) is never considered unconnected, even if there are no [SwConnectors](#) actually referring to it.]([RS_SWCT_00010](#), [RS_SWCT_03250](#), [RS_SWCT_03130](#))

Please note that [[TPS_SWCT_01573](#)] represents the immediate consequence of the semantics defined in [[TPS_SWCT_01112](#)].

[TPS_SWCT_01113] Connecting two [PortPrototypes](#) [Two [SwComponentPrototypes](#) are eventually connected by hooking up a [PPortPrototype](#) or [PRPortPrototype](#) of one [SwComponentPrototype](#) to a compatible [RPortPrototype](#) or [PRPortPrototype](#) of the other [SwComponentPrototypes](#). Please find more information concerning the definition of “compatibility” in section 6.]([RS_SWCT_03130](#), [RS_SWCT_03250](#))

Class	AbstractRequiredPortPrototype (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	This abstract class provides the ability to become a required PortPrototype.			
Base	ARObject, AtpBlueprintable, AtpFeature, AtpPrototype, Identifiable , Multilanguage Referrable, PortPrototype , Referrable			
Attribute	Datatype	Mul.	Kind	Note
requiredComSpec	RPortComSpec	*	aggr	Required communication attributes, one for each interface element.

Table 3.3: AbstractRequiredPortPrototype

Class	AbstractProvidedPortPrototype (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	This abstract class provides the ability to become a provided PortPrototype.			
Base	ARObject, AtpBlueprintable, AtpFeature, AtpPrototype, Identifiable , Multilanguage Referrable, PortPrototype , Referrable			
Attribute	Datatype	Mul.	Kind	Note
providedComSpec	PPortComSpec	*	aggr	Provided communication attributes per interface element (data element or operation).

Table 3.4: AbstractProvidedPortPrototype

Class	RPortPrototype			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	Component port requiring a certain port interface.			
Base	ARObject, AbstractRequiredPortPrototype , AtpBlueprintable, AtpFeature, AtpPrototype, Identifiable , MultilanguageReferrable, PortPrototype , Referrable			
Attribute	Datatype	Mul.	Kind	Note
requiredInterface	PortInterface	1	tref	The interface that this port requires, i.e. the port depends on another port providing the specified interface. Stereotypes: isOfType

Table 3.5: RPortPrototype

Class	PPortPrototype			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	Component port providing a certain port interface.			
Base	ARObject, AbstractProvidedPortPrototype , AtpBlueprintable, AtpFeature, AtpPrototype, Identifiable , MultilanguageReferrable, PortPrototype , Referrable			
Attribute	Datatype	Mul.	Kind	Note
providedInterface	PortInterface	1	tref	The interface that this port provides. Stereotypes: isOfType

Table 3.6: PPortPrototype

Class	PRPortPrototype			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	This kind of PortPrototype can take the role of both a required and a provided PortPrototype.			
Base	ARObject, AbstractProvidedPortPrototype , AbstractRequiredPortPrototype , AtpBlueprintable, AtpFeature, AtpPrototype, Identifiable , MultilanguageReferrable, PortPrototype , Referrable			
Attribute	Datatype	Mul.	Kind	Note
providedRequiredInterface	PortInterface	1	tref	This represents the PortInterface used to type the PRPortPrototype Stereotypes: isOfType

Table 3.7: PRPortPrototype

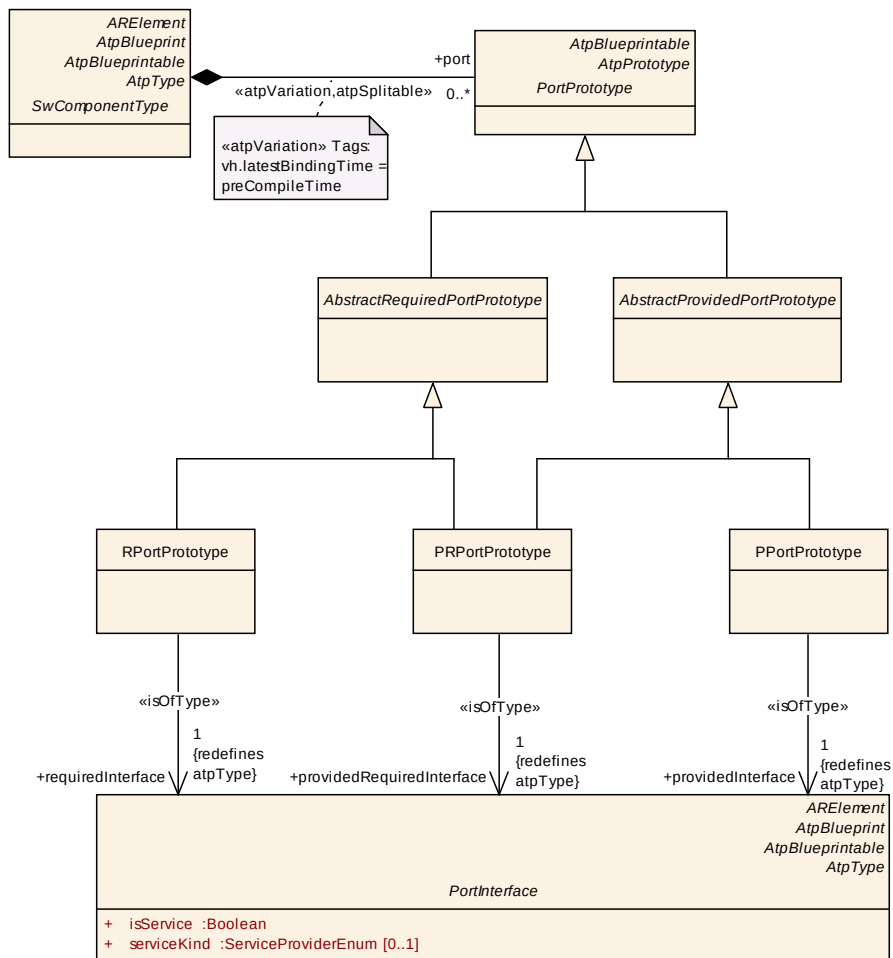


Figure 3.3: Components and Ports

[TPS_SWCT_01096] **PortGroup** [**PortPrototypes** can be logically grouped into **PortGroups**. This mechanism is used for implementing mode management features and further explained in chapter 4.6.] (RS_SWCT_03201)

3.2.3 AtomicSwComponentType

[TPS_SWCT_01108] **Added value of an AtomicSwComponentType** [As mentioned before, the term **AtomicSwComponentType** is a specific form of the general concept of the **SwComponentType**. The added value of an **AtomicSwComponentType** is that it can aggregate an **InternalBehavior** (see chapter 7).] (RS_SWCT_03040)

[TPS_SWCT_01109] **Adding the SwcInternalBehavior in a later process step** [The aggregation of **SwcInternalBehavior** is stereotyped <<atpSplitable>> to allow for adding the **SwcInternalBehavior** in a later process step. In other words, it is possible to completely develop the VFB view of a software-component and later add more details like **InternalBehavior**.] ()

Class	AtomicSwComponentType (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	An atomic software component is atomic in the sense that it cannot be further decomposed and distributed across multiple ECUs.			
Base	ARElement, ARObjekt, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable, SwComponentType			
Attribute	Datatype	Mul.	Kind	Note
internalBehavior	SwcInternalBehavior	0..1	aggr	The SwcInternalBehaviors owned by an AtomicSwComponentType can be located in a different physical file. Therefore the aggregation is «atpSplitable». Stereotypes: atpSplitable; atpVariation Tags: atp.Splitkey=internalBehavior, variationPoint.shortLabel vh.latestBindingTime=preCompileTime
symbolProps	SymbolProps	0..1	aggr	This represents the SymbolProps for the AtomicSwComponentType. Stereotypes: atpSplitable Tags: atp.Splitkey=shortName

Table 3.8: AtomicSwComponentType

There are several specialized [SwComponentTypes](#) to describe specific software-components used in the different parts of the AUTOSAR Layered Architecture [6]. Further details are mentioned in chapter 10 and 11.

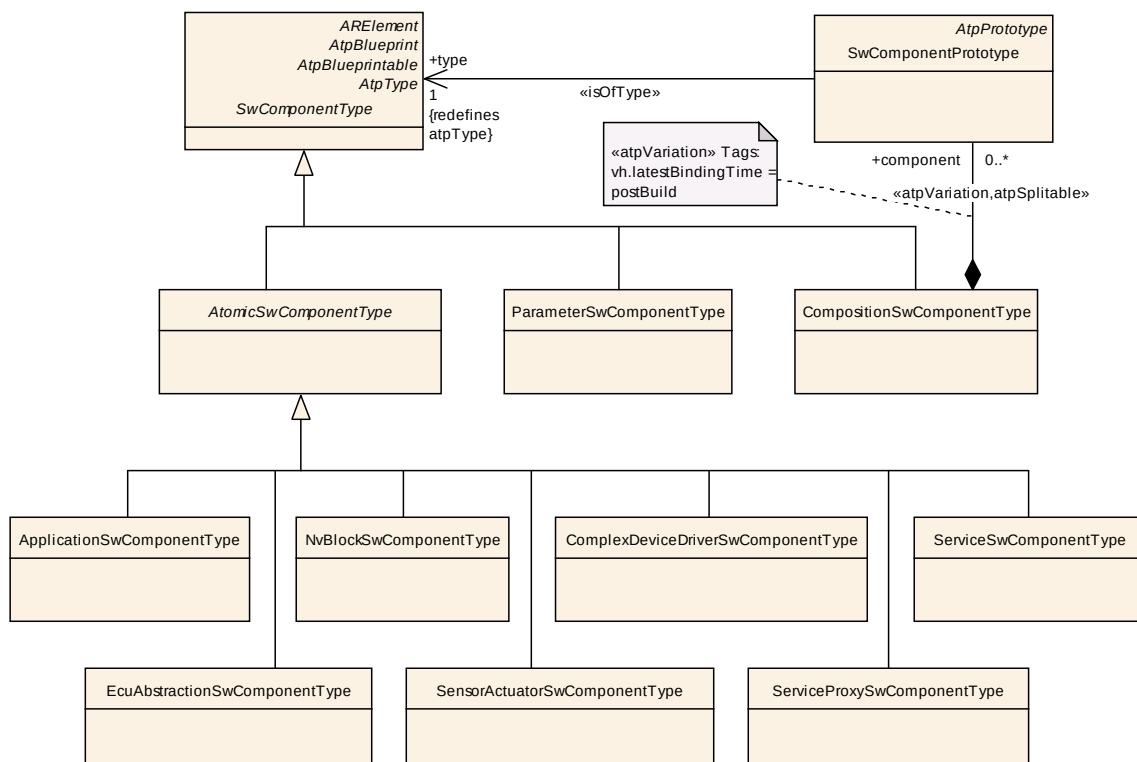


Figure 3.4: Overview of Component Types

The `ApplicationSwComponentType` is a specialization of `AtomicSwComponentType` for representing hardware-independent application software. The `ParameterSwComponentType` is a specialization of `SwComponentType` that can - in contrast to `AtomicSwComponentType` - not aggregate `SwcInternalBehavior`.

The purpose of the `NvBlockSwComponentType` is described in detail in section 11.5.2. The `ServiceSwComponentType` is described in section 11.3. Further on, the `EcuAbstractionSwComponentType` and the `ComplexDeviceDriverSwComponentType` are discussed in detail in section 10.

A description of the `ServiceProxySwComponentType` can be found in section 11.4 while the `SensorActuatorSwComponentType` is described in section 10.4.

Class	ApplicationSwComponentType			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	The ApplicationSwComponentType is used to represent the application software. Tags: atp.recommendedPackage=SwComponentTypes			
Base	ARElement, ARObject, AtomicSwComponentType, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable, SwComponentType			
Attribute	Datatype	Mul.	Kind	Note
–	–	–	–	–

Table 3.9: ApplicationSwComponentType

3.2.4 ParameterSwComponentType

[constr_1092] ParameterSwComponentType [A `ParameterSwComponentType` shall never aggregate a `SwcInternalBehavior` and also owns exclusively `PPortPrototypes` of type `ParameterInterface`.]()

However, a `ParameterSwComponentType` shall have the ability to aggregate `InstantiationDataDefProps`. By this means it is possible to define role-specific data properties of elements of composite data types used for the definition of calibration parameters in the scope of a `ParameterSwComponentType`.

For more information about this aspect please refer to section 7.5.4.

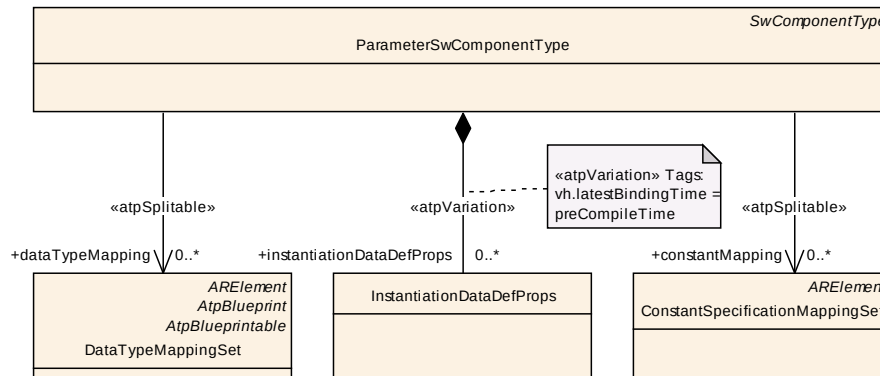


Figure 3.5: Details of **ParameterSwComponentType**

3.2.5 Symbolic Name of a Software-Component

Please note that an **AtomicSwComponentType** manifests itself in the source code of an RTE into which an instance of the **AtomicSwComponentType** is deployed. This implies potential naming conflicts if instances of **AtomicSwComponentType** that have identical **shortNames** are deployed into a specific RTE.

[TPS_SWCT_01110] Symbolic name of a software-component [To mitigate this potential hazard it is possible to provide the **AtomicSwComponentType** along with an accompanying symbolic name that can be used for resolving the name clash. The symbolic name is provided by means of the attribute **symbol** of the meta-class **SymbolProps** owned by **AtomicSwComponentType** in the role **symbolProps** (for more information, please refer to Figure 3.6).]()

Class	SymbolProps			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	This meta-class represents the ability to attach with the symbol attribute a symbolic name that is conform to C language requirements to another meta-class, e.g. AtomicSwComponentType , that is a potential subject to a name clash on the level of RTE source code.			
Base	ARObject, ImplementationProps , Referrable			
Attribute	Datatype	Mul.	Kind	Note
—	—	—	—	—

Table 3.10: **SymbolProps**

For more detailed information about how **SymbolProps** can be used to mitigate name clashes occurring during the integration of software-components on an AUTOSAR ECU, please refer to [4].

[TPS_SWCT_01000] Usage of attribute **symbol of the **symbolProps**** [In particular, the RTE generator shall take over the value of the attribute **symbol** of the **symbolProps** owned by a given **AtomicSwComponentType**. If and only if **symbolProps** is not defined the RTE generator shall take the **shortName** of the

`AtomicSwComponentType`. For the generation of symbols for `RunnableEntity`s [TPS_SWCT_01001] shall be observed.]()

[TPS_SWCT_01001] **Prefix symbols generated for the `RunnableEntity`** [If and only if the attribute `symbol` of a `symbolProps` owned by an `AtomicSwComponentType` exists, its value shall also be taken for prefixing the symbols generated for the `RunnableEntity`s owned by the `AtomicSwComponentType`.]()

Note: if `symbolProps` is not defined the behavior of the RTE generator is fully backwards compatible, i.e. existing implementations of `RunnableEntity`s do not have to be touched in order to conform with this version of the AUTOSAR standard.

This is a further measure to mitigate the risk of potential name clashes in the RTE code.

[TPS_SWCT_01635] **Naming conventions may support the effectiveness of `SymbolProps`** [Of course, there is a residual risk that even in the presence of `SymbolProps` name clashes may occur. Therefore, the definition of naming conventions may facilitate the avoidance of name clashes to the further degree.

However, these naming conventions can (with the support of the meta-model, e.g. by utilizing `SymbolProps` or `shortNamePattern`) still **only be defined on the model level**.](*RS_SWCT_00230*)

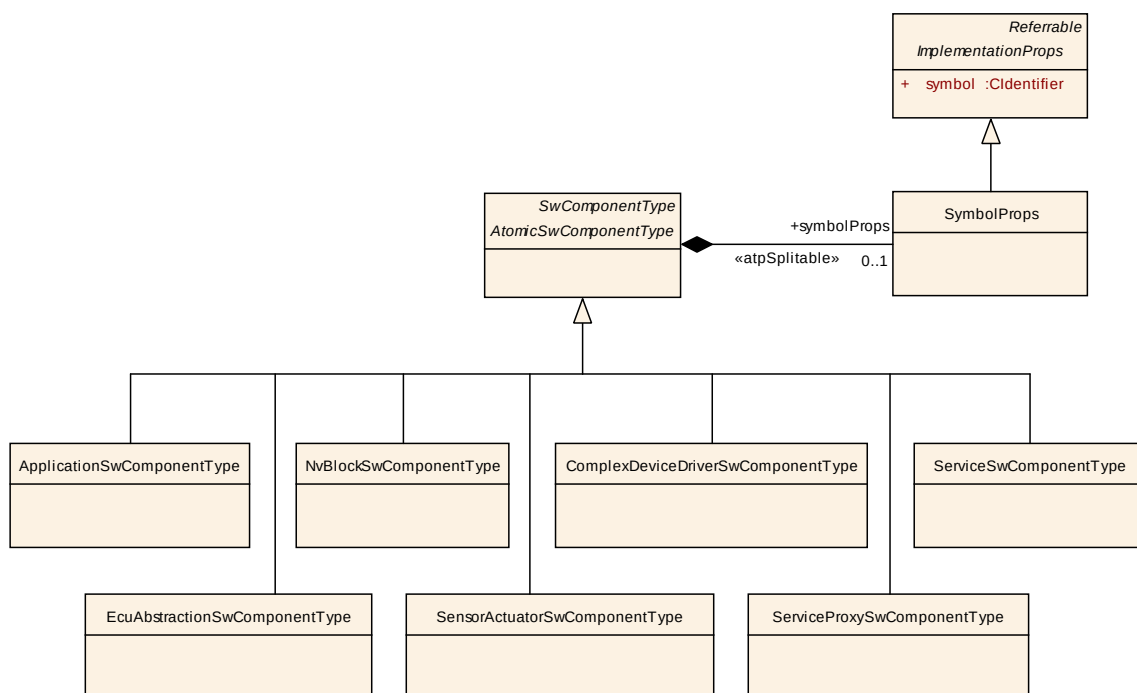


Figure 3.6: Overview of `AtomicSwComponentType`

3.3 Composition

3.3.1 Overview

[TPS_SWCT_01032] **CompositionSwComponentType** [The purpose of an AUTOSAR `CompositionSwComponentType` is to allow the encapsulation of specific functionality by aggregating existing software-components.](RS_SWCT_00190, RS_SWCT_02000, RS_SWCT_02020, RS_SWCT_03000)

[TPS_SWCT_01033] **Nested definition of CompositionSwComponentTypes** [Since a `CompositionSwComponentType` is also a `SwComponentType`, it again may be aggregated in further `CompositionSwComponentTypes`.](RS_SWCT_00190, RS_SWCT_02000, RS_SWCT_02020, RS_SWCT_03000)

This recursive relation is formally expressed in Figure 3.7.

It is important to understand that while compositions allow for (sub-) system abstraction, they are solely an *architectural element for the implementation of model scalability*. They simply group existing software-components and thereby take away complexity when viewing or designing logical software architecture.

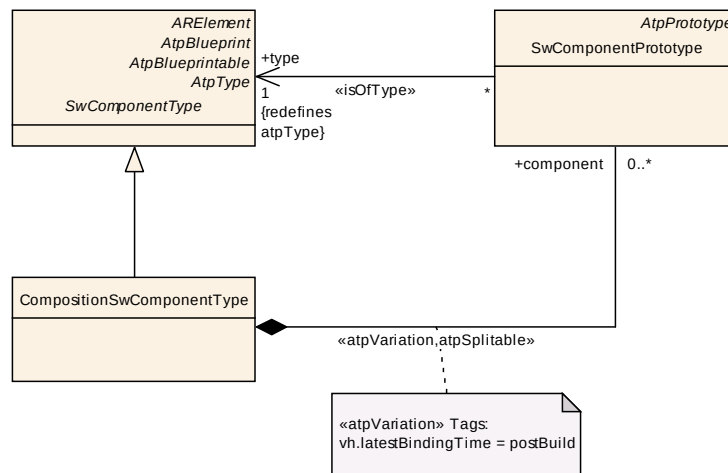


Figure 3.7: The recursive relation of software-components and compositions

Therefore, the definition of `CompositionSwComponentTypes` has no effect on how software-components interact with the Virtual Functional Bus (VFB). `CompositionSwComponentTypes` do not add any new functionality to what is already provided by the software-components they aggregate.

[TPS_SWCT_01034] **CompositionSwComponentTypes do not have any binary footprint** [As the main consequence, `CompositionSwComponentTypes` do not have any binary footprint in the ECU software.](RS_SWCT_00190, RS_SWCT_02000, RS_SWCT_02020, RS_SWCT_03000)

3.3.2 SwComponentPrototype

[TPS_SWCT_01035] **CompositionSwComponentType** aggregates **SwComponentPrototypes** [In terms of the AUTOSAR meta-model, a composition of software-components realized by the meta-class **CompositionSwComponentType** aggregates **SwComponentPrototypes** which in turn are typed by a **SwComponentType**.]([RS_SWCT_00190](#), [RS_SWCT_02000](#), [RS_SWCT_02020](#), [RS_SWCT_03000](#))

Please note that a **CompositionSwComponentType** is also a **SwComponentType**.

Class	CompositionSwComponentType			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Composition			
Note	A CompositionSwComponentType aggregates SwComponentPrototypes (that in turn are typed by SwComponentTypes) as well as SwConnectors for primarily connecting SwComponentPrototypes among each others and towards the surface of the CompositionSwComponentType. By this means hierarchical structures of software-components can be created. Tags: atp.recommendedPackage=SwComponentTypes			
Base	ARElement , ARObject , AtpBlueprint , AtpBlueprintable , AtpClassifier , AtpType , CollectableElement , Identifiable , MultilanguageReferrable , PackageableElement , Referrable , SwComponentType			
Attribute	Datatype	Mul.	Kind	Note
component	SwComponentPrototype	*	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 in that they are not scheduled by the RTE. The aggregation is marked as atpSplitable in order to allow the addition of service components to the ECU extract during the ECU integration. 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: atpSplitable; atpVariation Tags: atp.Splitkey=shortName, variation Point.shortLabel vh.latestBindingTime=postBuild

Attribute	Datatype	Mul.	Kind	Note
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. The aggregation is marked as atpSplitable in order to allow the extension of the ECU extract with AssemblySwConnectors between ApplicationSwComponentTypes and ServiceSwComponentTypes during the ECU integration. Stereotypes: atpSplitable; atpVariation Tags: atp.Splitkey=shortName, variation Point.shortLabel vh.latestBindingTime=postBuild
constantValueMapping	ConstantSpecificationMappingSet	*	ref	Reference to the ConstantSpecificationMapping to be applied for initValues of PPortComSpecs and RPortComSpec. Stereotypes: atpSplitable Tags: atp.Splitkey=constantValueMapping
dataTypeMapping	DataTypeMappingSet	*	ref	Reference to the DataTypeMapping 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 implementers 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. Stereotypes: atpSplitable Tags: atp.Splitkey=dataTypeMapping

Attribute	Datatype	Mul.	Kind	Note
instantiationRTEEventProps	InstantiationRTEEventProps	*	aggr	This allows to define instantiation specific properties for RTE Events, in particular for instance specific scheduling. Stereotypes: atpSplittable; atpVariation Tags: atp.Splitkey=shortLabel, variationPoint.shortLabel vh.latestBindingTime=codeGenerationTime

Table 3.11: CompositionSwComponentType

Class	SwComponentPrototype			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Composition			
Note	Role of a software component within a composition.			
Base	ARObject, AtpFeature, AtpPrototype, Identifiable , MultilanguageReferrable, Referrable			
Attribute	Datatype	Mul.	Kind	Note
type	SwComponentType	1	tref	Type of the instance. Stereotypes: isOfType

Table 3.12: SwComponentPrototype

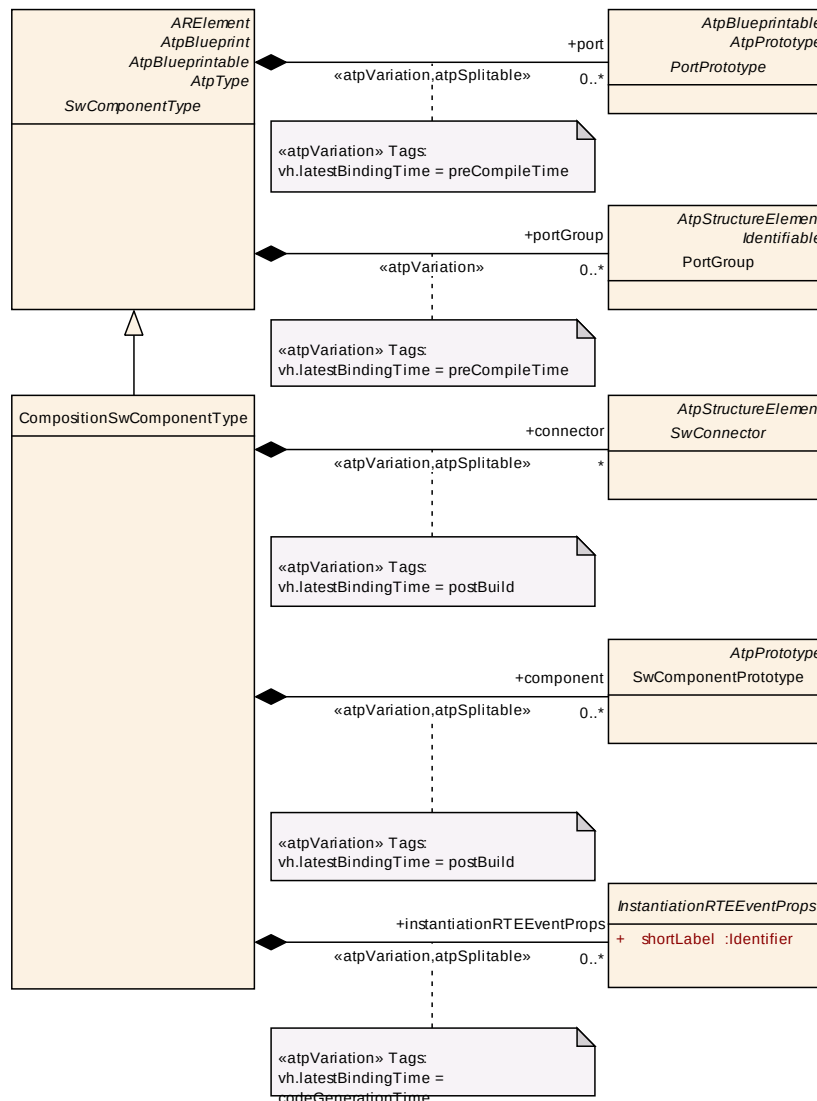


Figure 3.8: Composition and the meta-classes aggregated

[TPS_SWCT_01036] **SwComponentPrototype** implements a specific role [Therefore, a **SwComponentPrototype** implements the usage of a **SwComponentType** in a specific role.] ([RS_SWCT_00190](#), [RS_SWCT_02000](#), [RS_SWCT_02020](#), [RS_SWCT_03000](#))

[TPS_SWCT_01037] arbitrary numbers of **SwComponentPrototypes** can be created [In general, arbitrary numbers of **SwComponentPrototypes** that refer to specific **SwComponentTypes** can be created.] ([RS_SWCT_00190](#), [RS_SWCT_02000](#), [RS_SWCT_02020](#), [RS_SWCT_03000](#))

Example: a **SwComponentPrototype** “LeftDoorControl” fulfills the role of implementing the **SwComponentType** “DoorControl” for the left door of a vehicle while the **SwComponentPrototype** “RightDoorControl” fulfills the role of the **SwComponentType** “DoorControl” for the right door.

[TPS_SWCT_01080] **Delegation ports** [Note that being a **SwComponentType**, a **CompositionSwComponentType** also exposes **PortPrototypes** to the out-

side world. However, the `PortPrototypes` are only delegated and do not play the same role as `PortPrototypes` attached to `AtomicSwComponentTypes`.
]([RS_SWCT_03130](#))

[TPS_SWCT_01081] Implications of being a delegation port [Being a `PortPrototype` attached to a `CompositionSwComponentType` has the following implications:

- The delegation has to follow the rules defined in chapter 6.
- By creating `PortPrototypes` on the surface of a specific `CompositionSwComponentType` it is explicitly decided whether or not the contents of an “inner” port contained in the `CompositionSwComponentType` is exposed to the outside world.

]([RS_SWCT_03130](#))

Please note that the semantics of the delegation of `PortPrototypes` are similar to encapsulation mechanisms like public and private members in object-oriented programming languages.

One implication of the concept of `CompositionSwComponentType` is that the application software of an entire vehicle eventually is represented by one `CompositionSwComponentType`. This so-called top-level composition has a special role in the context of the AUTOSAR System Template [11].

However, please note that a top-level composition might have (unconnected) `PortPrototypes` in order to allow for reuse as part of another system.

[constr_1035] Recursive definition of `CompositionSwComponentType` [The recursive definition of a `CompositionSwComponentType` that eventually contains a `SwComponentPrototype` typed by the same `CompositionSwComponentType` shall not be feasible.]()

3.3.3 Connectors

[TPS_SWCT_01079] `SwConnector` [Note that `CompositionSwComponentType` also aggregates the abstract meta-class `SwConnector` for connecting the `SwComponentPrototypes` contained among each other (see Figure 3.8).
]([RS_SWCT_03130](#))

`CompositionSwComponentTypes` contain two kinds of `SwConnectors`:

- **[TPS_SWCT_01082] `AssemblySwConnector`** [`AssemblySwConnectors` interconnect `PortPrototypes` of `SwComponentPrototypes` that are part of the `CompositionSwComponentType`.]([RS_SWCT_03130](#))
- **[TPS_SWCT_01083] `DelegationSwConnector`** [`DelegationSwConnectors` connect from “inner” `PortPrototypes` to delegated “outer” `PortPrototypes`.]([RS_SWCT_03130](#))

[constr_1032] DelegationSwConnector can only connect PortPrototypes of the same kind [A `DelegationSwConnector` can only connect `PortPrototypes` of the same kind, i.e. `PPortPrototype` to `PPortPrototype` and `RPortPrototype` to `RPortPrototype`.]()

[TPS_SWCT_01084] Outer PortPrototype is referenced by multiple DelegationSwConnectors [In the case that an outer `PortPrototype` is referenced by multiple `DelegationSwConnectors` the semantic is the multiplication of the `AssemblySwConnectors` referencing the outer `PortPrototypes`.](*RS_SWCT_03130*)

[constr_1086] SwConnector between two specific PortPrototypes [Each pair of `PortPrototypes` can only be connected by one and only one `SwConnector`.]()

In other words, it is not supported to create two different `SwConnectors` that connect the same pair of `PortPrototypes`.

[TPS_SWCT_01638] Existence of SwConnector between two PRPortPrototypes [**[constr_1086]** applies also in the case that two `PRPortPrototypes` are connected with each other. In particular, the roles

- `AssemblySwConnector.requester`
- `AssemblySwConnector.provider`
- `PassThroughSwConnector.providedOuterPort`
- `PassThroughSwConnector.requiredOuterPort`

do **not** establish a direction in this case.]()

For clarification, **[TPS_SWCT_01638]** means that the `SwConnector` represents the ability for bi-directional communication between the two `PRPortPrototypes`.

[constr_1087] AssemblySwConnector inside CompositionSwComponentType [An `AssemblySwConnector` can only connect `PortPrototypes` of `SwComponentPrototypes` that are owned by the same `CompositionSwComponentType`]()

[constr_1088] DelegationSwConnector inside CompositionSwComponentType [A `DelegationSwConnector` can only connect a `PortPrototype` of a `SwComponentPrototype` that is owned by the same `CompositionSwComponentType` that also owns the connected delegation `PortPrototype`.]()

In the context of attaching a `DelegationSwConnector` to an inner `PRPortPrototype` there is some ambiguity to be considered. In particular, from the formal point of view it would be feasible to use **either** a `PPortInCompositionInstanceRef` **or** a `RPortInCompositionInstanceRef`.

The ability to use one or the other meta-class arbitrarily is considered confusing. Therefore, **[TPS_SWCT_01515]** has been defined to remove the unnecessary degree of freedom.

[TPS_SWCT_01515] **PPortInCompositionInstanceRef** shall be used for attaching **DelegationSwConnector** to an inner **PRPortPrototype** [For the implementation of the attachment of a **DelegationSwConnector** to an inner **PRPortPrototype** the meta-class **PPortInCompositionInstanceRef** shall be used.]()

[constr_1100] **Unconnected RPortPrototype** typed by a **DataInterface** [For any element in an unconnected **RPortPrototype** typed by a **DataInterface** there shall be a **requiredComSpec** that defines an **initValue**.]()

Class	SwConnector (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Composition			
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 , Multilanguage Referrable, Referrable			
Attribute	Datatype	Mul.	Kind	Note
mapping	PortInterfaceMapping	0..1	ref	Reference to a PortInterfaceMapping specifying the mapping of unequal named PortInterface elements of the two different PortInterfaces typing the two PortPrototypes which are referenced by the ConnectorPrototype.

Table 3.13: SwConnector

Class	AssemblySwConnector			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Composition			
Note	AssemblySwConnectors are exclusively used to connect SwComponentPrototypes in the context of a CompositionSwComponentType.			
Base	ARObject, AtpClassifier, AtpFeature, AtpStructureElement, Identifiable , Multilanguage Referrable, Referrable , SwConnector			
Attribute	Datatype	Mul.	Kind	Note
provider	AbstractProvidedPortPrototype	0..1	iref	Instance of providing port.
requester	AbstractRequiredPortPrototype	0..1	iref	Instance of requiring port.

Table 3.14: AssemblySwConnector

Class	DelegationSwConnector			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Composition			
Note	A delegation connector delegates one inner PortPrototype (a port of a component that is used inside the composition) to a outer PortPrototype of compatible type that belongs directly to the composition (a port that is owned by the composition).			
Base	ARObject, AtpClassifier, AtpFeature, AtpStructureElement, Identifiable , Multilanguage Referrable, Referrable , SwConnector			
Attribute	Datatype	Mul.	Kind	Note

Attribute	Datatype	Mul.	Kind	Note
innerPort	PortPrototype	1	iref	The port that belongs to the ComponentPrototype in the composition Tags: xml.typeElement=true
outerPort	PortPrototype	1	ref	The port that is located on the outside of the CompositionType

Table 3.15: DelegationSwConnector

One specific use case for the application of [SwConnectors](#) is exemplified by the figures 3.9 and 3.11. A specific [CompositionSwComponentType](#) exists in two variants where one (more complex) variant foresees the existence of a [SwComponentPrototype](#) inside the [CompositionSwComponentType](#) (depicted by 3.9) and the other (because it is implementing a simpler semantics) does not need the [SwComponentPrototype](#).

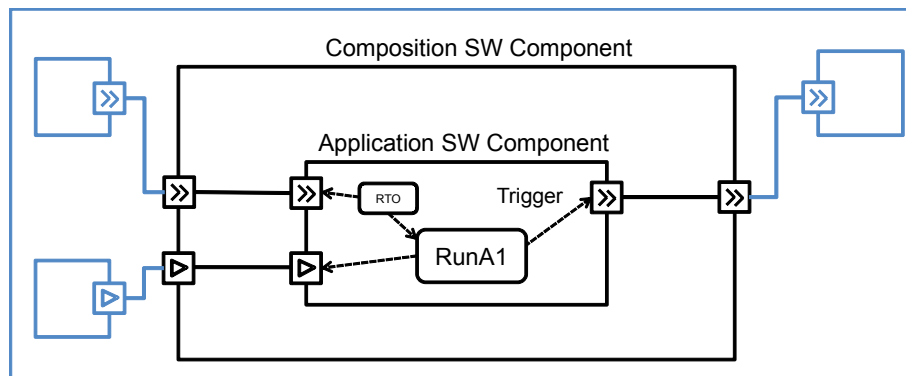


Figure 3.9: Use case for [PassThroughSwConnector](#) (I)

Class	PassThroughSwConnector			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Composition			
Note	This kind of SwConnector can be used inside a CompositionSwComponentType to connect two delegation PortPrototypes.			
Base	ARObject, AtpClassifier, AtpFeature, AtpStructureElement, Identifiable , Multilanguage Referrable, Referrable , SwConnector			
Attribute	Datatype	Mul.	Kind	Note
providedOuterPort	AbstractProvidedPortPrototype	1	ref	This represents the provided outer delegation PortPrototype of the PassThroughSwConnector.
requiredOuterPort	AbstractRequiredPortPrototype	1	ref	This represents the required outer delegation PortPrototype of the PassThroughSwConnector.

Table 3.16: PassThroughSwConnector

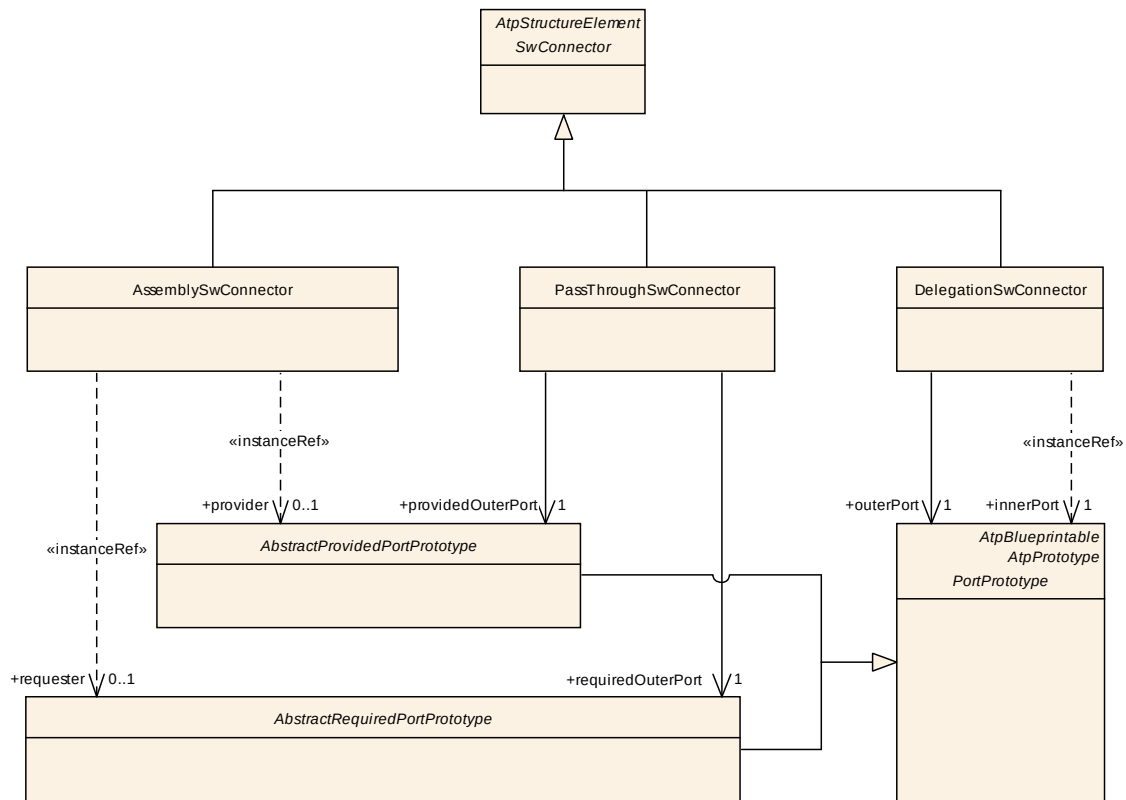


Figure 3.10: Connectors

Without the ability to define a `PassThroughSwConnector` the second variant could only be implemented by defining a dummy `SwComponentPrototype` inside the `CompositionSwComponentType`. However, the dummy `SwComponentPrototype` would need to define `RunnableEntity`s that are created for the sole purpose of being able to shovel the data from (e.g. for sender-receiver communication) `RPortPrototypes` to `PPortPrototypes`.

This would not only be cumbersome it would also obviously require additional resources (memory and code) at run-time. Plus, the existence of addition `RunnableEntity`s also unnecessarily increases the propagation delay of information flowing around inside the ECU.

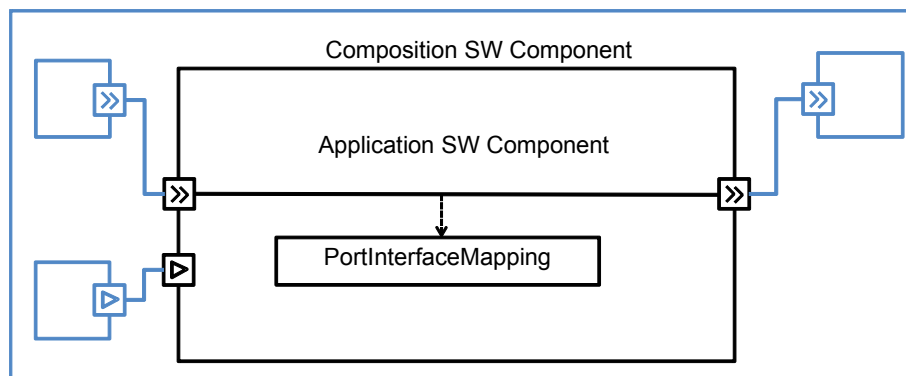


Figure 3.11: Use case for `PassThroughSwConnector` (II)

[TPS_SWCT_01507] The role of `PassThroughSwConnector` [`PassThroughSwConnector` can be taken to connect `PortPrototypes` owned by the same `CompositionSwComponentType`. In other words, `PassThroughSwConnector` creates a bypass inside a `CompositionSwComponentType` from the `requiredOuterPort` to the `providedOuterPort` (or vice versa) without involving `SwComponentPrototypes`.]()

[constr_1252] Creation of a loop involving a `PassThroughSwConnector` is not allowed [A `PassThroughSwConnector` is not allowed if the required outer `PortPrototype` is directly or indirectly connected to the provided outer `PortPrototype` without the placement of a `SwComponentPrototype` typed by an `AtomicSwComponentType` in the chain of `SwConnectors`.]()

In other words, according to [constr_1252] it is not allowed to create a “infinite loop” by means of a `PassThroughSwConnector` and at least one `AssemblySwConnector` that connects the `requiredOuterPort` to the `providedOuterPort`.

3.3.4 Instantiation-specific RTEEvents

[TPS_SWCT_02507] Instantiation-specific `RTEEvents` [It is possible to specify instantiation specific properties of an `RTEEvent` by applying `InstantiationRTEEventProps` in the role `instantiationRteEventProps`.

This allows to use the same `ApplicationSwComponentType` in different timing scenarios. Even if the scheduling is an issue of the `SwcInternalBehavior`, the instance specific definition of timing needs to be specified on the level of a `CompositionSwComponentType`.](*RS_SWCT_03046*, *RS_SWCT_03270*)

As an example for [TPS_SWCT_02507], please consider a software-component that implements a closed-loop control algorithm.

This software-component can potentially be deployed to “slow” and “fast” control scenarios. As the actual time-base of the control algorithm is derived from the scheduling implemented in the RTE it obviously facilitates the overall design if the timing can be defined on “instance” level.

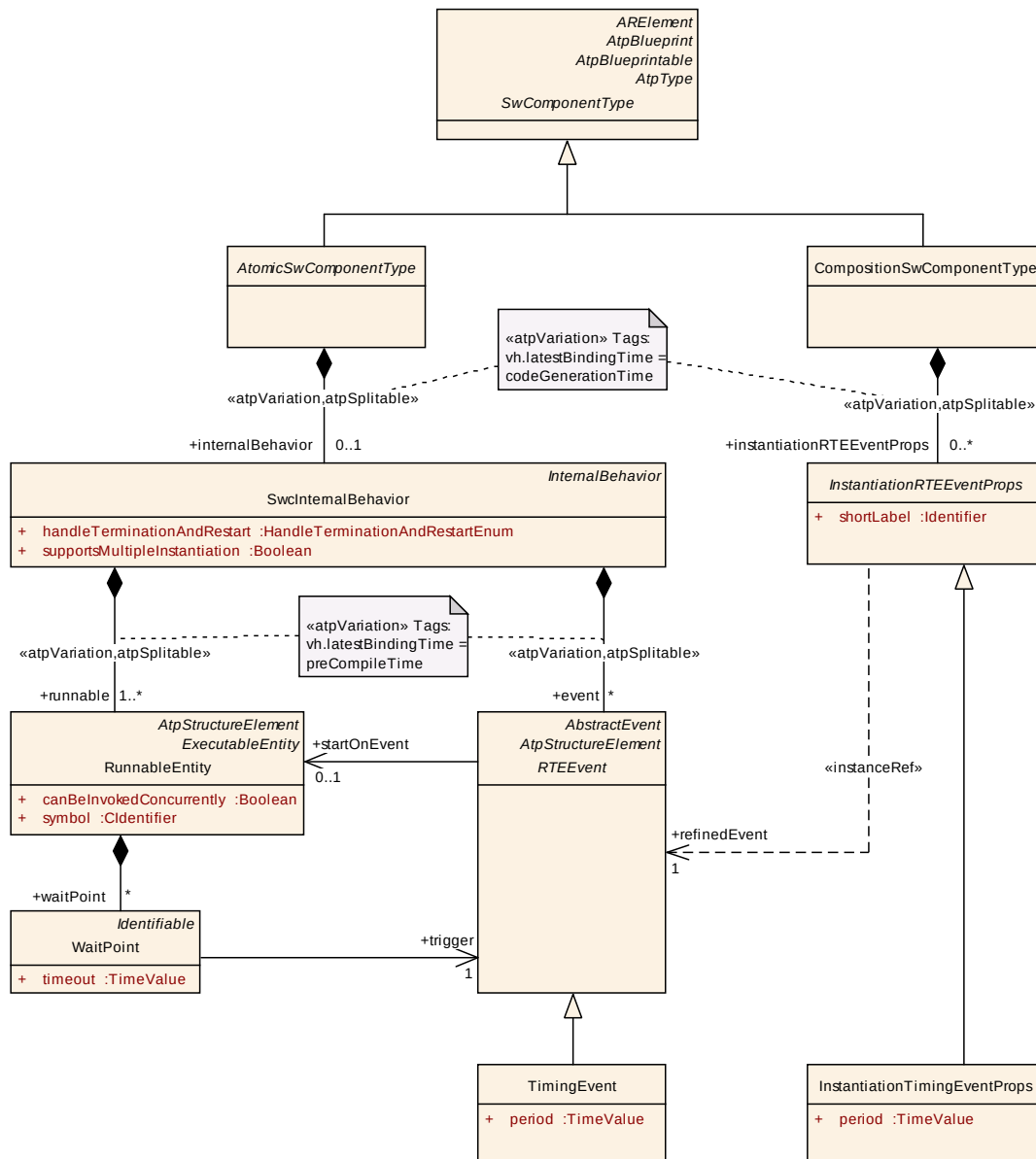


Figure 3.12: Instantiation specific Properties of RTEEvents

Class	InstantiationRTEEventProps (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Composition			
Note	This meta class represents the ability to refine the properties of RTEEvents for particular instances of a software component.			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
refinedEvent	RTEEvent	1	iref	This instance ref denotes the Timing Event for which the period shall be refined on an instance level.
shortLabel	Identifier	1	ref	The main purpose of the shortLabel is to contribute to the splitkey of aggregations that are «atpSplitable».

Table 3.17: InstantiationRTEEventProps

[constr_1233] **InstantiationTimingEventProps** shall only reference **TimingEvent** [An **InstantiationTimingEventProps** shall only reference **TimingEvent** in the role **refinedEvent**. A reference to other kinds of **RTEEvents** is not supported.]()

3.4 Port Interface

[TPS_SWCT_01025] **The role of PortPrototypes in the AUTOSAR architecture** [A **PortPrototype** mainly contributes the functionality of being a *connection point* to the AUTOSAR concept.

The details, i.e. with respect to what kind of information is actually transported between two **PortPrototypes** is defined by the **PortInterface**.]([RS_SWCT_00010](#), [RS_SWCT_00080](#), [RS_SWCT_00110](#), [RS_SWCT_02030](#), [RS_SWCT_03010](#))

[TPS_SWCT_01026] **The role of PortInterfaces in the AUTOSAR architecture** [**PortInterfaces** (see Figure 3.14) are used to support a design-by-contract workflow, i.e. a **PortInterface** provides means to formally verify structural and dynamic compatibility between software-components.]([RS_SWCT_00010](#), [RS_SWCT_00080](#), [RS_SWCT_00110](#), [RS_SWCT_02030](#), [RS_SWCT_03010](#))

In other words: **PortInterfaces** represent a pivotal point in the AUTOSAR concept.

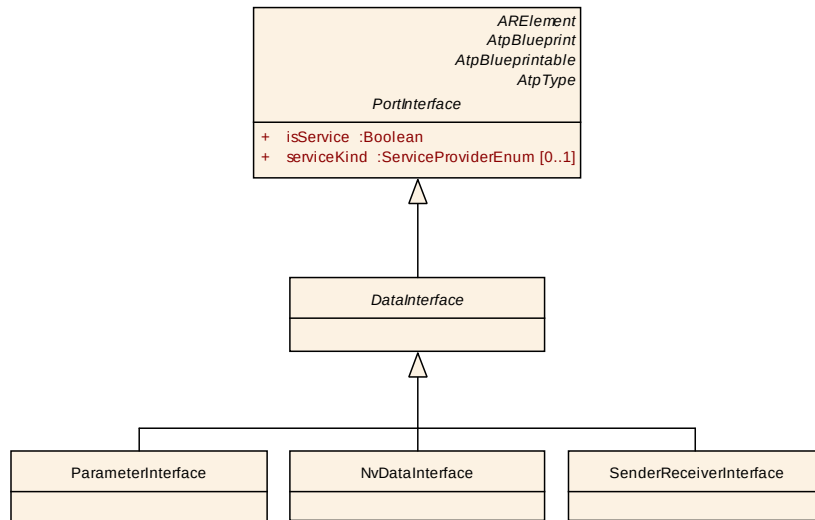
Please note that a **PortInterface** creates a name space for the information contained. This allows for defining the details of a specific **PortInterface** without having to care for possible side-effects on other **PortInterfaces**. Again, this property of the AUTOSAR concept directly supports re-usability.

[TPS_SWCT_01027] **Different flavors of PortInterfaces** [Within the AUTOSAR concept, different flavors of **PortInterfaces** are defined:

- **SenderReceiverInterface**
- **NvDataInterface**
- **ParameterInterface**
- **ModeSwitchInterface**
- **ClientServerInterface**
- **TriggerInterface**

]([RS_SWCT_00010](#), [RS_SWCT_00080](#), [RS_SWCT_00110](#), [RS_SWCT_02030](#))

[TPS_SWCT_01069] **DataInterface** is defined as abstract base class [Please note that the conceptual relationship of **SenderReceiverInterface**, **NvDataInterface**, and **ParameterInterface** is expressed by the definition of the abstract base class **DataInterface**.]([RS_SWCT_00010](#), [RS_SWCT_00080](#), [RS_SWCT_00110](#), [RS_SWCT_03010](#))

Figure 3.13: **DataInterface** as an abstract base class

Please find more details about the specialization of the **PortInterface** concept in chapter 4.2.3 and 4.2.2.

Class	PortInterface (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
Note	Abstract base class for an interface that is either provided or required by a port of a software component.			
Base	ARElement, ARObjekt, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Attribute	Datatype	Mul.	Kind	Note
isService	Boolean	1	attr	This flag is set if the PortInterface is to be used for communication between an • ApplicationSwComponentType or • ServiceProxySwComponentType or • SensorActuatorSwComponentType or • ComplexDeviceDriverSwComponentType • ServiceSwComponentType • EcuAbstractionSwComponentType and a ServiceSwComponentType (namely an AUTOSAR Service) located on the same ECU. Otherwise the flag is not set.
serviceKind	ServiceProviderEnum	0..1	attr	This attribute provides further details about the nature of the applied service.

Table 3.18: **PortInterface**

Class	DataInterface (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
Note	The purpose of this meta-class is to act as an abstract base class for subclasses that share the semantics of being concerned about data (as opposed to e.g. operations).			
Base	ARElement, ARObject, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, PortInterface, Referrable			
Attribute	Datatype	Mul.	Kind	Note
–	–	–	–	–

Table 3.19: DataInterface

[TPS_SWCT_01070] **PortInterface** acts as a *type* for a **PortPrototype** [From an abstract point of view, a **PortInterface** acts as a *type* for a **PortPrototype**. This means in particular that several **PortPrototypes** can be typed by the same **PortInterface**.] ([RS_SWCT_00010](#), [RS_SWCT_00080](#), [RS_SWCT_00110](#), [RS_SWCT_03010](#))

Of course, this aspect facilitates the creation of valid connections between software-components dramatically. By using a specific **PortInterface** for typing particular **PortPrototypes** the latter are eligible for being connected to each other by definition.

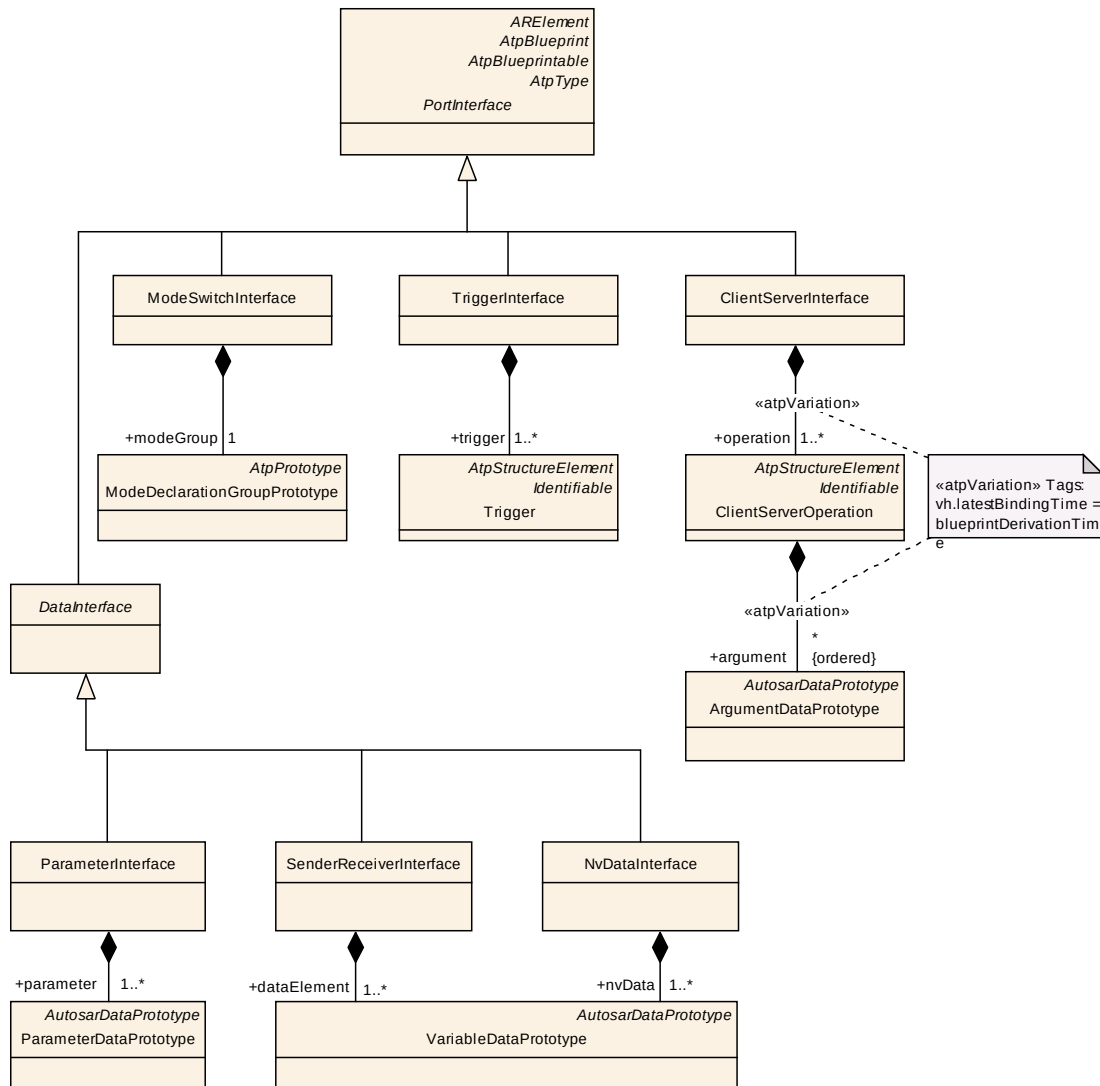


Figure 3.14: **PortInterfaces** in the AUTOSAR meta-model

However, the creation of a valid connection does not need to be based on the usage of identical **PortInterfaces**. It is also possible to use different, but *compatible* **PortInterfaces**. The details about compatibility of **PortInterfaces** are described in chapter 6.

[constr_1036] Connect kinds of **PortInterfaces** [It shall not be possible to connect **PortPrototypes** typed by **PortInterfaces** of different kinds. Subclasses of **DataInterface** make an exception from this rule and can be used for creating connections to each other.]()

For clarification, a connection between a **PortPrototype** typed by a **SenderReceiverInterface** and a **PortPrototype** typed by a **ClientServerInterface** shall not be possible. However, the creation of a connection between a **PortPrototype** typed by a **SenderReceiverInterface** and a **PortPrototype** typed by a **ParameterInterface** is supported.

[constr_1137] Applicability of `ParameterInterface` [A `PPortPrototype` typed by a `ParameterInterface` can **only** be owned by a `ParameterSwComponentType`.]()

Please note that `PortInterfaces` also play an important role in the context of defining so-called AUTOSAR services. In particular, by means of the attribute `isService` a `PortInterface` can define whether or not it is supposed to be used in the context of an AUTOSAR service and in addition to this it may define (by means of the attribute `serviceKind`) what kind of service is intended.

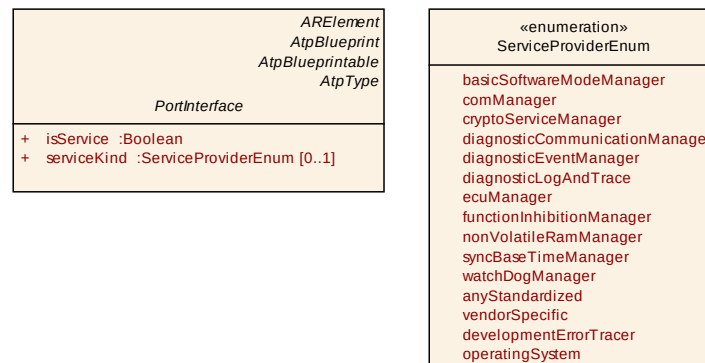


Figure 3.15: `PortInterfaces` and AUTOSAR services

The information contained in `serviceKind` can be used in various ways. The primary intent is to distinguish between the usage of standardized AUTOSAR services from the usage of a vendor-specific service. This information may have an impact on the development- and build process of software-components that use the `PortInterface`.

In addition, it is also possible to use the information contained in `serviceKind` for filtering the presentation of an AUTOSAR model in an AUTOSAR authoring tool and e.g. display the nature of the service `PortPrototypes` independently of the content of the corresponding `PortInterface`.

[TPS_SWCT_01003] Inconsistencies regarding the value of `serviceKind` and the actual implementation of the `PortInterface` [In case of inconsistencies between the value of `serviceKind` and the actual implementation of the `PortInterface` the implementation of the `PortInterface` wins over the value of attribute `PortInterface.serviceKind` (which, for the intended purpose shall be considered an annotation rather than a semantically binding information).]()

[TPS_SWCT_01004] Default value if `serviceKind` is not defined [if the attribute `serviceKind` is not defined in the context of a specific `PortInterface` the default value `anyStandardized` shall be assumed.]()

[constr_1174] `PortInterfaces` used in the context of `CompositionSwComponentTypes` cannot refer to AUTOSAR services [`CompositionSwComponentTypes` shall not own `PortPrototypes` typed by `PortInterfaces` where the attribute `isService` is set to `true`.]()

Enumeration	ServiceProviderEnum
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds
Note	This represents a list of possible service providers
Literal	Description
anyStandardized	This value means that the specific nature is either unknown or it is not important for the given purpose. This is also the default value for any attribute of type ServiceProviderEnum
basicSoftwareModeManager	The service relates to the Basic Software Mode Manager (BswM)
comManager	The service relates to the COM Manager (ComM).
cryptoServiceManager	The service relates to the Crypto Service Manager (CsM).
developmentErrorTracer	The service relates to the Development Error Tracer (DET).
diagnosticCommunicationManager	The service relates to the Diagnostic Communication Manager (DCM).
diagnosticEventManager	The service relates to the Diagnostic Event Manager (DEM).
diagnosticLogAndTrace	The service relates to the Diagnostic Log and Trace (DLT).
ecuManager	The service relates to the ECU Manager (EcuM).
functionInhibitionManager	The service relates to the Function Inhibition Manager (FIM).
nonVolatileRamManager	The service relates to the Non-Volatile RAM Manager (NvM).
operatingSystem	The service relates to the Operating System (OS).
syncBaseTimeManager	The service relates to the Sync Time Base Manager (StbM).
vendorSpecific	This value denotes a vendor-specific service.
watchDogManager	The service relates to the Watchdog Manager (WdgM).

Table 3.20: ServiceProviderEnum

[TPS_SWCT_01005] Usage of [SwcServiceDependency](#)s for vendor-specific services [[SwcServiceDependency](#)s can also be used for vendor-specific services. In this case the [SwcServiceDependency](#) shall not contain any of the standardized [ServiceNeeds](#).]()

Please find more details about the relation of [PortInterfaces](#) to AUTOSAR services in chapter 11.

4 Details: Software Components, Ports, and Interfaces

4.1 Introduction

The specification of the Virtual Functional Bus (VFB) [3] explains the main communication paradigms for communication among software-components: *client/server* for operation-based communication, and *sender/receiver* for data-based communication.

The nature of the two communication paradigms is quite different, and so is the modeling of [SenderReceiverInterfaces](#) and [ClientServerInterfaces](#) and their related meta-classes.

[TPS_SWCT_01516] [PortInterface](#) describes the static structure of information interchange [[PortInterfaces](#) are limited to the description of the static structure of the exchanged information; the dynamic attributes (please refer to chapter 4.5) relevant for communication are attached to [PortPrototypes](#).] ([RS_SWCT_00010](#), [RS_SWCT_00080](#), [RS_SWCT_00110](#), [RS_SWCT_02030](#), [RS_SWCT_03010](#))

4.2 Port Interface Details

4.2.1 Introduction

The usage of value encodings (for more information please refer to section 5.2.6) is limited within the context of [PortInterfaces](#).

[constr_1045] Supported value encodings for [SwBaseType](#) in the context of [PortInterfaces](#) [The supported value encodings for the usage within a [Port-Interface](#) are:

- 2C: Two's complement
- IEEE754: floating point numbers
- ISO-8859-1: ASCII-Strings
- ISO-8859-2: ASCII-Strings
- WINDOWS-1252: ASCII-Strings
- UTF-8: UCS Transformation Format 8
- [UTF-16](#): Character encoding for Unicode *code points* based on 16 bit *code units* [16]
- UCS-2: Universal Character Set 2
- NONE: Unsigned Integer

- **BOOLEAN:** This represents an integer to be interpreted as boolean.

⌋()

[constr_1046] Applicability of [constr_1045] ⌈ [constr_1045] applies **only** if the value of the attribute `isService` is set to `false`. ⌋()

[constr_1295] PortInterfaces and category DATA_REFERENCE ⌈ A `DataPrototype` defined in the context of a `PortInterface` used by an `ApplicationSwComponentType` or `SensorActuatorSwComponentType` that is (after potential indirections via `TYPE_REFERENCE` are resolved) either typed by or mapped to an `ImplementationDataType` of category `DATA_REFERENCE` shall only be used if either the provider or the requester of the information represents a `ServiceSwComponentType`, a `ComplexDeviceDriverSwComponentType`, a `ParameterSwComponentType`, or an `NvBlockSwComponentType`, or the `EcuAbstractionSwComponentType`. ⌋()

Note: [constr_1295] corresponds to [SWS_RTE_07670].

4.2.2 Sender Receiver Communication

[TPS_SWCT_01114] SenderReceiverInterface ⌈ `SenderReceiverInterfaces` allow for the specification of the typically asynchronous communication pattern where a sender provides data that is required by one or more receivers.

While the actual communication takes place via the respective `PortPrototypes`, a `SenderReceiverInterface` allows for formally describing what kind of information is sent and received. ⌋()

Class	SenderReceiverInterface			
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
Note	A sender/receiver interface declares a number of data elements to be sent and received. Tags: atp.recommendedPackage=PortInterfaces			
Base	<code>ARElement</code> , <code>ARObject</code> , <code>AtpBlueprint</code> , <code>AtpBlueprintable</code> , <code>AtpClassifier</code> , <code>AtpType</code> , <code>CollectableElement</code> , <code>DataInterface</code> , <code>Identifiable</code> , <code>MultilanguageReferrable</code> , <code>PackageableElement</code> , <code>PortInterface</code> , <code>Referrable</code>			
Attribute	Datatype	Mul.	Kind	Note
dataElement	<code>VariableDataPrototype</code>	1..*	aggr	The data elements of this <code>SenderReceiverInterface</code> .
invalidationPolicy	<code>InvalidationPolicy</code>	*	aggr	InvalidationPolicy for a particular <code>dataElement</code>

Table 4.1: SenderReceiverInterface

Class	InvalidationPolicy			
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
Note	Specifies whether the component can actively invalidate a particular dataElement. If no invalidationPolicy points to a dataElement this is considered to yield the identical result as if the handleInvalid attribute was set to dontInvalidate.			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
dataElement	VariableDataPrototype	1	ref	Reference to the dataElement for which the InvalidationPolicy applies.
handleInvalid	HandleInvalidEnum	0..1	attr	This attribute controls how invalidation is applied to the dataElement.

Table 4.2: InvalidationPolicy

Enumeration	HandleInvalidEnum
Package	M2::AUTOSARTemplates::SWComponentTemplate::Communication
Note	Strategies of handling the reception of invalidValue.
Literal	Description
dontInvalidate	Invalidation is switched off.
externalReplacement	Replace a received invalidValue. The replacement value is sourced from the externalReplacement.
keep	The application software is supposed to handle signal invalidation on RTE API level either by DataReceiveErrorEvent or check of error code on read access.
replace	Replace a received invalidValue. The replacement value is specified by the initValue.

Table 4.3: HandleInvalidEnum

A [SenderReceiverInterface](#) focuses on the description of information items represented by [VariableDataPrototypes](#) (see section 5.3).

A [VariableDataPrototype](#) aggregated in the role of [dataElement](#) represents an atomic¹ piece of information transmitted among [PortPrototypes](#) typed by a [SenderReceiverInterface](#).

[TPS_SWCT_01115] [invalidationPolicy](#) [An [invalidationPolicy](#) specifies whether the sending component can actively invalidate a particular [dataElement](#) and which strategy of handling the reception of [invalidValue](#) on the receiver side shall be implemented.]()

Further information about the related concept of an [invalidValue](#) is provided in chapter 5.4.2

¹Note that the term “atomic” does not have any implication on the implementation on a concrete computing platform

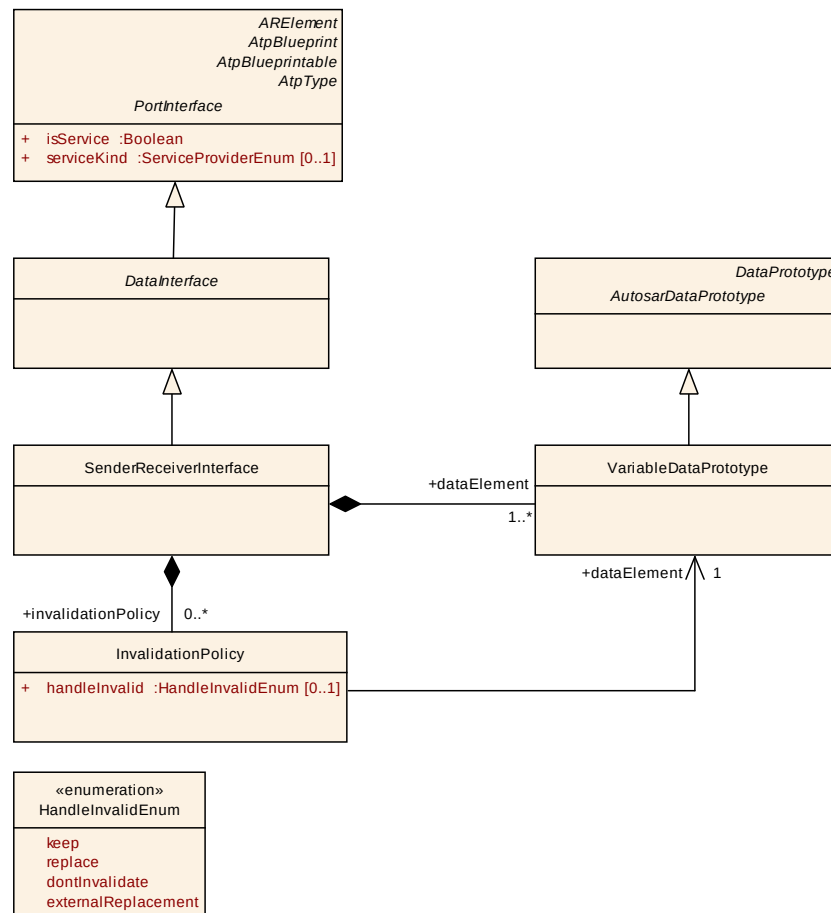


Figure 4.1: dataElements of a SenderReceiverInterface

Note that a `SenderReceiverInterface` provides a name space for the definition of `VariableDataPrototypes`. In terms of the AUTOSAR meta-model this aspect is indicated by the inheritance relation to `DataPrototype` (which in turn inherits from `Identifiable`). Please find more information on the creation of name spaces in [12].

[TPS_SWCT_01116] swImplPolicy [The `swImplPolicy` (see section 5.4) indicates the way how a `VariableDataPrototype` shall be processed at the receiver's side. If set to `queued` the semantics is that the corresponding `VariableDataPrototype` needs to be added to a *queue* (or in other words: a FIFO data structure) from which it is later consumed by the actual receiver software-component.]()

[constr_1200] Queued communication is not applicable for dataElements owned by PRPortPrototype [The `swImplPolicy` shall not be set to `queued` for any `dataElement` owned by a `PRPortPrototype`.]()

[TPS_SWCT_01176] last-is-best semantics for sender-receiver communication [If `swImplPolicy` is set to any other valid value of `SwImplPolicyEnum` then *last is best* semantics applies.]()

Please note that the definition of `VariableDataPrototype` may possibly come very close to the reader's idea of a *signal*. However, different kinds of signals have a specific

meaning in the AUTOSAR concept, especially in the context of the AUTOSAR System Template [11].

[TPS_SWCT_01117] Communication patterns for sender-receiver communication [[PortPrototypes](#) typed by a [SenderReceiverInterface](#) may be connected to establish a 1:n (i.e. one sender, multiple receivers) communication relationship. It is also possible to establish a n:1 (i.e. many senders, one receiver) communication pattern.]()

[constr_1033] Communication scenarios for sender/receiver communication [For sender/receiver communication, it is not allowed to create a communication scenario where n sender are connected to m receivers where m and n are **both** greater than 1.]()

Factually, [\[constr_1033\]](#) is not applicable to a scenario where several [PRPortPrototypes](#) are connected by a chain of [AssemblySwConnectors](#) or [PassThroughSwConnectors](#).

[constr_1202] Supported connections by [AssemblySwConnector](#) for [PortPrototypes](#) typed by a [SenderReceiverInterface](#) or [NvDataInterface](#) [For the modeling of [AssemblySwConnectors](#) between [PortPrototypes](#) typed by a [SenderReceiverInterface](#) or [NvDataInterface](#), **only** the connections documented in Table 4.4 are supported by AUTOSAR.]()

	RPortPrototype	PPortPrototype	PRPortPrototype
RPortPrototype	No	Yes	Yes
PPortPrototype	Yes	No	Yes
PRPortPrototype	Yes	Yes	Yes

Table 4.4: Supported connections for [PortPrototypes](#) typed by a [SenderReceiverInterface](#) or [NvDataInterface](#)

[constr_1203] Supported connections by [DelegationSwConnector](#) for [PortPrototypes](#) typed by a [SenderReceiverInterface](#) or [NvDataInterface](#) [For the modeling of [DelegationSwConnectors](#) between [PortPrototypes](#) typed by a [SenderReceiverInterface](#) or [NvDataInterface](#), **only** the connections documented in Table 4.5 are supported by AUTOSAR.]()

innerPort	outerPort		
	RPortPrototype	PPortPrototype	PRPortPrototype
RPortPrototype	Yes	No	Yes
PPortPrototype	No	Yes	Yes
PRPortPrototype	Yes	Yes	Yes

Table 4.5: Supported connections for [PortPrototypes](#) typed by a [SenderReceiverInterface](#) or [NvDataInterface](#)

4.2.3 Client Server Communication

The underlying semantics of a client/server communication is that a client may initiate the execution of an operation by a server that supports the operation.

The server executes the operation and, when completed, it provides the client with the result (synchronous operation call) or else the client checks for the completion of the operation by itself (asynchronous operation call).

[constr_1037] Client shall not be connected to multiple servers [A client shall not be connected to multiple servers such that an operation call would be handled by more than one server.]()

4.2.3.1 Client Server Interface

A [ClientServerInterface](#), to some extent, is a counterpart to the [Sender-ReceiverInterface](#)².

Instead of defining pieces of information to be transferred among software-components, a [ClientServerInterface](#) defines a collection of [ClientServer-Operations](#).

Class	ClientServerInterface			
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
Note	A client/server interface declares a number of operations that can be invoked on a server by a client. Tags: atp.recommendedPackage=PortInterfaces			
Base	ARElement , ARObject , AtpBlueprint , AtpBlueprintable , AtpClassifier , AtpType , CollectableElement , Identifiable , MultilanguageReferrable , PackageableElement , PortInterface , Referrable			
Attribute	Datatype	Mul.	Kind	Note
operation	ClientServerOperation	1..*	aggr	ClientServerOperation(s) of this ClientServerInterface. Stereotypes: atpVariation Tags: vh.latestBindingTime=blueprintDerivationTime
possibleError	ApplicationError	*	aggr	Application errors that are defined as part of this interface.

Table 4.6: ClientServerInterface

²However, different connection patterns apply, see [\[constr_1037\]](#)

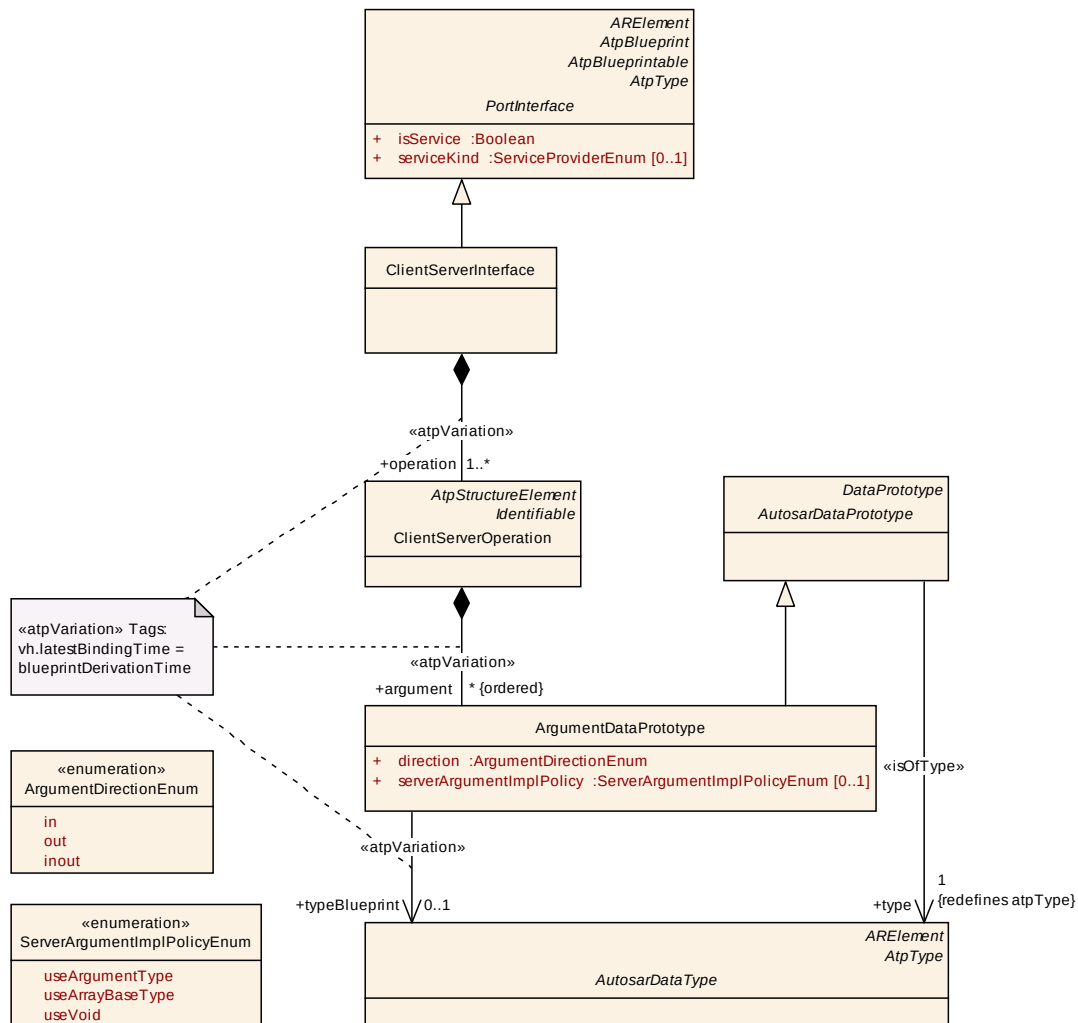


Figure 4.2: ClientServerOperations of a ClientServerInterface

[TPS_SWCT_01118] **ClientServerInterface** [As depicted in Figure 4.2, a **ClientServerInterface** is composed of **ClientServerOperations**, i.e. a **ClientServerOperation** cannot be reused in the context of a different **ClientServerInterface** |()

[TPS_SWCT_01106] **ClientServerOperation** [A ClientServerOperation consists of 0..* **ArgumentDataPrototypes**. The latter may be

- passed to the operation (i.e. the direction is “in”)
- passed to and returned from the operation (i.e. the direction is “inout”)
- returned from the operation (i.e. the direction is “out”)

The aggregation represents a variation point. | (RS SWCT 03141)

Class	ClientServerOperation			
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
Note	An operation declared within the scope of a client/server interface.			
Base	ARObject, AtpClassifier, AtpFeature, AtpStructureElement, Identifiable , Multilanguage Referrable, Referrable			
Attribute	Datatype	Mul.	Kind	Note
argument (ordered)	ArgumentDataPrototype	*	aggr	An argument of this ClientServerOperation Stereotypes: atpVariation Tags: vh.latestBindingTime=blueprintDerivationTime
possibleError	ApplicationError	*	ref	Possible errors that may be raised by the referring operation.

Table 4.7: ClientServerOperation

Class	ArgumentDataPrototype			
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
Note	An argument of an operation, much like a data element, but also carries direction information and is owned by a particular ClientServerOperation.			
Base	ARObject, AtpFeature, AtpPrototype, AutosarDataPrototype , DataPrototype , Identifiable , MultilanguageReferrable, Referrable			
Attribute	Datatype	Mul.	Kind	Note
direction	ArgumentDirectionEnum	1	attr	This attribute specifies the direction of the argument prototype.
serverArgumentImplPolicy	ServerArgumentImplPolicyEnum	0..1	attr	This defines how the argument type of the servers RunnableEntity is implemented. If the attribute is not defined this has the same semantics as if the attribute is set to the value useArgumentType for primitive arguments and structures and to the value useArrayBaseType for arrays.
typeBlueprint	AutosarDataPrototype	0..1	ref	This allows to denote the intended type within blueprints. It shall be replaced by a proper type when deriving Interfaces from the Blueprint. Stereotypes: atpVariation Tags: vh.latestBindingTime=blueprintDerivationTime

Table 4.8: ArgumentDataPrototype

[TPS_SWCT_01119] Direction of [ArgumentDataPrototypes](#) [To cover these cases, [ArgumentDataPrototype](#) defines an attribute *direction*, possible values are *in* (pass to operation), *out* (return from operation), and *inout* (pass to and return from operation).]()

In many common programming languages (like C), an operation is yet another data type. This makes it for example possible to pass a reference to an operation as an argument to another operation.

This is *not* allowed in the AUTOSAR concept.

[TPS_SWCT_01517] *ClientServerOperation* cannot be passed as a reference [It is not possible to pass a reference to a *ClientServerOperation* as an *ArgumentDataPrototype* in another *ClientServerOperation*.]()

Essentially, all *ArgumentDataPrototypes* in a *ClientServerOperation* can be passed (conceptually) by value (from the client to the server and/or from the server to the client depending on the *direction* of the *ArgumentDataPrototype*).

[TPS_SWCT_01120] Client needs to provide *ArgumentDataPrototypes* [When the client invokes an operation, it needs to provide a value for each *ArgumentDataPrototype* that is of direction *in* or *inout*.]()

[TPS_SWCT_01121] Pass correct data type [The value passed to an *ArgumentDataPrototype* of direction *in* or *inout* needs to be of the corresponding *Datatype*.]()

[TPS_SWCT_01122] Synchronous call of *ClientServerOperation* [In the case of synchronous operation call, the client expects to receive a response to the invocation of the operation.

As part of the response, it receives a value (of the correct *AutosarDataType*) for each *ArgumentDataPrototype* that is of direction *out* or *inout*.]()

<i>Enumeration</i>	ArgumentDirectionEnum
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::Primitive Types
Note	Use cases: - Arguments in <i>ClientServerOperation</i> can have different directions that need to be formally indicated because they have an impact on how the function signature looks like eventually. - Arguments in <i>BswModuleEntry</i> already determine a function signature, but the direction is used to specify the semantics, especially of pointer arguments.
Literal	Description
in	The argument value is passed to the callee.
inout	The argument value is passed to the callee but also passed back from the callee to the caller.
out	The argument value is passed from the callee to the caller.

Table 4.9: ArgumentDirectionEnum

Each `ClientServerOperation` provides a name space for its `ArgumentDataPrototypes` and therefore has a unique identifier which identifies the operation within the corresponding `ClientServerInterface`.

The `ClientServerOperations` have no ordering within a `ClientServerInterface` (there is no such thing as the “first” operation)³.

[TPS_SWCT_01123] No default values for `ArgumentDataPrototypes` [It is not possible to define default values for `ArgumentDataPrototypes` defined in the context of a `ClientServerOperation`. Default values might lead to complicated mappings to programming languages.]()

[TPS_SWCT_01124] Definition of `ArgumentDataPrototypes` within the context of a `ClientServerOperation` is ordered [In contrast to the unordered relationship of `ClientServerInterface` to `ClientServerOperation`, the definition of `ArgumentDataPrototypes` within the context of a `ClientServerOperation` is ordered, i.e. a `ClientServerOperation` may have a *first* argument⁴.]()

Please note that `ArgumentDataPrototype` inherits from `AutosarDataPrototype` and therefore has a reference to a concrete `AutosarDataType`.

The RTE Generator uses the referred `AutosarDataTypes` to determine the data types of the arguments depending on the value of the attribute `ArgumentDataPrototype.serverArgumentImplPolicy`.

<i>Enumeration</i>	ServerArgumentImplPolicyEnum
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface
Note	This defines how the argument type of the servers <code>RunnableEntity</code> is implemented.
Literal	Description
useArgumentType	The argument type of the <code>RunnableEntity</code> is derived from the <code>AutosarDataType</code> of the <code>ArgumentPrototype</code> .
useArrayBaseType	The argument type of the <code>RunnableEntity</code> is derived from the <code>AutosarDataType</code> of the elements of the array that corresponds to the <code>ArgumentPrototype</code> . This represents the base type of the array in C.
useVoid	The argument type of the <code>RunnableEntity</code> is void.

Table 4.10: ServerArgumentImplPolicyEnum

³In different parts of the definition of a `ClientServerInterface`, a “calling-order” of the `ClientServerOperations` might be prescribed: the client might be required to use the `ClientServerOperations` in a certain logical ordering.

However, this ordering has nothing to do with the order in which the `ClientServerOperations` are listed in the definition of a `ClientServerInterface`

⁴ Giving the `ArgumentDataPrototypes` of a `ClientServerOperation` both an ordering and a unique identifier might seem redundant.

For example, in the operation “foo(a, b, c)”, we can refer to the “second argument” or to “the argument named b”. In many common programming languages (like C or Java), only the *ordering* is actually used by the client during the invocation of the server (the client invokes the operation as “foo(1,2,3)” not as “foo(a=1,c=3,b=2)”.

In addition, the names of the arguments represent an arbitrary choice made when implementing of the invocation. In C, only the data types and ordering of the arguments constitute the signature, *not* the names of the arguments.

Please note that the scenario described in [TPS_SWCT_01125] is depicted in Figure 4.3.

[TPS_SWCT_01125] `serverArgumentImplPolicy` [The option `useArrayBaseType` is intended to implement “Server Runnables” which are able to handle array typed arguments of different length. In this case the software component does have several Server Ports.

At least one argument of the `ClientServerOperations` is typed by `AutosarDataTypes` of `category` `ARRAY` but the length of the arrays defined by `ApplicationArrayElement.maxNumberOfElements` respectively `arraySize` might be different for the individual Server Ports.

All `ClientServerOperations` in the `PortPrototypes` are triggering the same `RunnableEntity` with `OperationInvokedEvents`.

If the `serverArgumentImplPolicy` is set to `useArrayBaseType` the RTE Generator does not require the compatibility of `ClientServerOperations` for such `ArgumentDataPrototypes` and uses the `baseType` of the array as the argument’s data type instead of the array data type with a particular length.

The option `useVoid` is available to implement Server `RunnableEntities` which are able to handle arbitrary typed arguments of different length - typically of `category` `STRUCTURE`. The design of the software component implementing the server is similar as explained for `useArrayBaseType`.

If the `serverArgumentImplPolicy` is set to `useVoid` the RTE Generator does not require the compatibility of `ClientServerOperations` for such `ArgumentDataPrototypes` and uses `void` as the argument data type. `]()`

[constr_1297] Applicability of `serverArgumentImplPolicy` set to `useArrayBaseType` [The value of the attribute `ArgumentDataPrototype.serverArgumentImplPolicy` shall only be set to `useArrayBaseType` for an `ArgumentDataPrototype` that is typed by an `AutosarDataType` that is (after all `TYPE_REFERENCES` are resolved) either an `ImplementationDataType` of `category` `ARRAY` or an `ApplicationDataType` mapped to (after all `TYPE_REFERENCES` are resolved) an `ImplementationDataType` of `category` `ARRAY`. `]()`

[constr_1286] `serverArgumentImplPolicy` and `ArgumentDataPrototype` typed by primitive data types [The value of the attribute `ArgumentDataPrototype.serverArgumentImplPolicy` shall **not** be set to `useVoid` for an `ArgumentDataPrototype` of `direction` `in` that is typed by an `AutosarDataType` that boils down to a primitive C data type (see [TPS_SWCT_01565]). `]()`

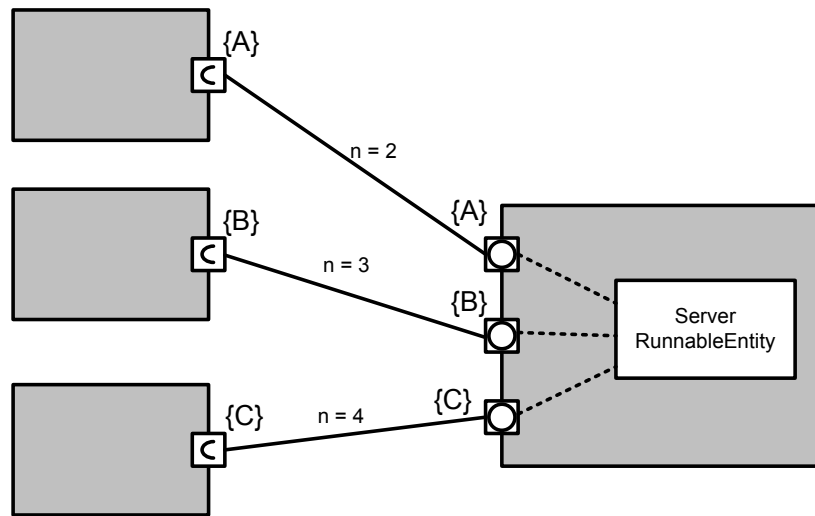


Figure 4.3: Example for [TPS_SWCT_01125]

Please note that the server `RunnableEntity` needs information about the currently used array length respectively structure size by usage of additionally arguments passed by the Client or via `PortDefinedArgumentValue`.

Note further that a `ClientServerInterface` does not define any timing information (how quickly the client expects a response of the server). It does not define how the threading works (if the client for example blocks until the response comes back from the server).

It also does not define explicitly how information is passed between an implementation of the client and the server and the underlying RTE (for example: through “pointers” or “by value”).

[constr_1204] Supported connections by `AssemblySwConnector` for `Port-Prototypes` typed by a `ClientServerInterface`, `ModeSwitchInterface`, or `TriggerInterface` [For the modeling of `AssemblySwConnectors` between `PortPrototypes` typed by a `ClientServerInterface`, `ModeSwitchInterface`, or `TriggerInterface`, **only** the connections documented in Table 4.11 are supported by AUTOSAR.]()

	RPortPrototype	PPortPrototype	PRPortPrototype
RPortPrototype	No	Yes	Yes
PPortPrototype	Yes	No	No
PRPortPrototype	Yes	No	No

Table 4.11: Supported connections for `PortPrototypes` typed by a `ClientServerInterface`, `ModeSwitchInterface`, or `TriggerInterface`

[constr_1205] Supported connections by `DelegationSwConnector` for `PortPrototypes` typed by a `ClientServerInterface`, `ModeSwitchInterface`, or `TriggerInterface` [For the modeling of `DelegationSwConnectors` between `PortPrototypes` typed by a `ClientServerInterface`, `ModeSwitchInterface`, or `TriggerInterface`, **only** the connections documented in Table 4.12 are supported by AUTOSAR.]()

innerPort	outerPort		
	RPortPrototype	PPortPrototype	PRPortPrototype
RPortPrototype	Yes	No	No
PPortPrototype	No	Yes	No
PRPortPrototype	No	Yes	No

Table 4.12: Supported connections for `PortPrototypes` typed by a `ClientServerInterface`, `ModeSwitchInterface`, or `TriggerInterface`

4.2.3.2 Error Handling in Client/Server Communication

This section describes the handling of errors occurring either within an application software-component or during the communication across the VFB [3]. Errors that are created and consumed by basic software modules are not in the scope of this document and therefore will not be discussed.

Therefore, errors in the scope of this document are divided into two simple classes:

- infrastructure errors and
- application errors.

A software-component implementation uses RTE API methods to communicate with other software-components. During this communication certain errors can occur as a result of infrastructure faults, like a bus is not working, or an expected data value was not arriving in time.

These errors are listed in the RTE specification [2], as they are an inherent feature of the infrastructure provided by the VFB. Software-components will therefore typically not raise infrastructure errors on their own.

Instead, AUTOSAR the basic software and the RTE will determine infrastructure faults and communicate the corresponding error codes to the relevant software-components.

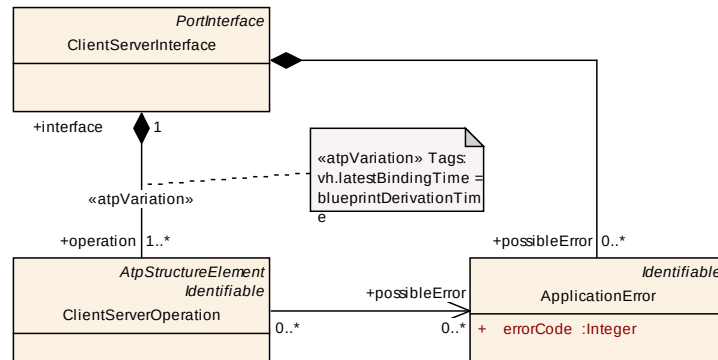


Figure 4.4: Application error meta-model

[TPS_SWCT_01491] **AUTOSAR system does not need to explicitly describe infrastructure errors** [As the fixed set of infrastructure errors is defined as an implicit part of the VFB, a developer of an AUTOSAR system does not need to explicitly describe these.

It is assumed that these might occur at run-time and application developers should take measures to handle them.]()

Application errors, on the other hand, are specific to the functionality or information that is described in form of a **PortInterface**. It is not possible to define such errors up front, instead they are defined at design time of a certain **PortInterface**.

In principle, such **ApplicationErrors** could be part of all kinds of **PortInterfaces**

[TPS_SWCT_01490] **AUTOSAR supports ApplicationErrors only for ClientServerInterfaces** [As of now, AUTOSAR supports (as depicted by Figure 4.4) **ApplicationErrors** only for **ClientServerInterfaces**.]()

[constr_1102] **ApplicationError in the scope of one SwComponentType** [A **SwComponentType** may have **PortPrototypes** typed by different **PortInterfaces** with equal **shortName** but conflicting **ApplicationErrors**.

ApplicationErrors are considered conflicting if **ApplicationErrors** with the same **shortName** do have different **errorCodes**.]()

[constr_1108] **Value of ApplicationError.errorCode** [The value of **ApplicationError.errorCode** shall not exceed the closed interval 1 .. 63. The following exception applies: **only** in case **possibleError** is supposed to represent **E_OK** the value 0 shall be allowed.]()

By [constr_1108] it is possible to ensure that only the six least significant bits of a return value shall be used for indicating an application error.

Class	ApplicationError			
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
Note	This is a user-defined error that is associated with an element of an AUTOSAR interface. It is specific for the particular functionality or service provided by the AUTOSAR software component.			
Base	ARObject, Identifiable , MultilanguageReferrable, Referrable			
Attribute	Datatype	Mul.	Kind	Note
errorCode	Integer	1	attr	The RTE generator is forced to assign this value to the corresponding error symbol. Note that for error codes certain ranges are predefined (see RTE specification).

Table 4.13: ApplicationError

Consequently, [ClientServerOperations](#) may be associated with a number of [ApplicationErrors](#) they possibly raise. These errors are defined as part of the [ClientServerInterface](#).

[constr_1038] Reference to [ApplicationError](#) [A [possibleError](#) referenced by a [ClientServerOperation](#) shall be owned by the [ClientServerInterface](#) that also owns the [ClientServerOperation](#).]()

4.2.4 External Trigger Event Communication

[TPS_SWCT_01196] Semantics of an external trigger event communication [The underlying semantics of an external trigger event communication is that a trigger source may initiate the execution of [RunnableEntity](#)s in the connected trigger sinks. Typically (but not necessarily) these [RunnableEntity](#)s are executed in a sequential order.]()

[TPS_SWCT_01197] [TriggerInterface](#) [The [TriggerInterface](#) defines a set of [Trigger](#) to be communicated between software-components. The Trigger represents a special kind of events at which occurrence the trigger sinks shall react in a particular manner.]()

Class	TriggerInterface			
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
Note	A trigger interface declares a number of triggers that can be sent by an trigger source. Tags: atp.recommendedPackage=PortInterfaces			
Base	ARElement , ARObject, AtpBlueprint , AtpBlueprintable, AtpClassifier, AtpType, CollectableElement, Identifiable , MultilanguageReferrable, PackageableElement, PortInterface , Referrable			
Attribute	Datatype	Mul.	Kind	Note
trigger	Trigger	1..*	aggr	The Trigger of this trigger interface.

Table 4.14: TriggerInterface

Class	Trigger			
Package	M2::AUTOSARTemplates::CommonStructure::TriggerDeclaration			
Note	A trigger which is provided (i.e. released) or required (i.e. used to activate something) in the given context.			
Base	ARObject, AtpClassifier, AtpFeature, AtpStructureElement, Identifiable , Multilanguage Referrable, Referrable			
Attribute	Datatype	Mul.	Kind	Note
swImplPolicy	SwImplPolicyEnum	0..1	attr	This attribute, when set to value queued, allows for a queued processing of Triggers.
triggerPeriod	MultidimensionalTime	0..1	aggr	Optional definition of a period in case of a periodically (time or angle) driven external trigger.

Table 4.15: Trigger

Class	MultidimensionalTime			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::MultidimensionalTime			
Note	This is used to specify a multidimensional time value based on ASAM CSE codes. It is specified by a code which defined the basis of the time and a scaling factor which finally determines the time value. If for example the cseCode is 100 and the cseCodeFactor is 360, it represents 360 angular degrees. If the cseCode is 2 and the cseCodeFactor is 50 it represents 50 microseconds			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
cseCode	CseCodeType	1	attr	Specifies the time base by means of CSE codes.
cseCodeFactor	Integer	1	attr	The scaling factor for the time value based on the specified CSE code.

Table 4.16: MultidimensionalTime

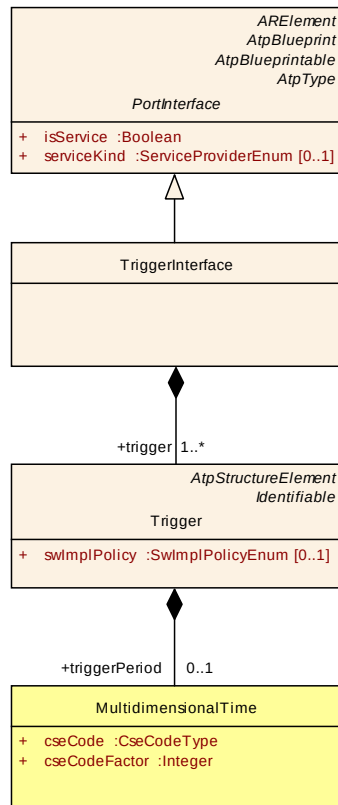


Figure 4.5: Trigger of a TriggerInterface

As illustrated in Figure 4.5, a [TriggerInterface](#) is composed of [Trigger](#).

[TPS_SWCT_01198] Period for periodic triggering [A [Trigger](#) can optionally define a period for periodic triggering. It is expressed via the meta-class [MultidimensionalTime](#) in terms of time or angle. Note that the main use case for this is to specify the properties if the trigger is coming from the Basic Software e.g. from a Complex Driver, it is not used as an input for the RTE generator.]()

Apart from this, a [TriggerInterface](#) does not define any timing information (e.g. how quickly the source expects a reaction of the sinks). This is property of the timing information in the templates.

[constr_1104] Trigger sink and trigger source [An [RPortPrototype](#) typed by a [TriggerInterface](#) shall not be referenced by more than one [SwConnectors](#) that are in turn referencing [PPortPrototypes](#) typed by [TriggerInterfaces](#) that contain [Triggers](#) with the same [shortName](#).]()

[\[constr_1104\]](#) boils down to the requirement that trigger communication shall not be implemented in a n:1 scenario.

To be clear, the n:1 scenario is not supported for trigger communication because there is no active use case for it. Support would require the implementation of queue management for Trigger communication.

[TPS_SWCT_01199] Queued processing of Triggers [It may happen that at least tentatively a [Trigger](#) source fires [Triggers](#) faster than they can be processed on

the side of the `Trigger` sink. To support this use case it is possible to process trigger event communication in a queued manner.

In this case the `Triggers` are added to a queue from where the foremost trigger is dequeued and processed when the processing of the current `Trigger` is done. Please note that the queue size is **not** subject to definition in the scope of this document. The actual queue size is defined during the process of RTE configuration.

The specification of whether or not a `Trigger` is subject to queued processing is controlled by the attribute `Trigger.swImplPolicy`. `]()`

[constr_1169] Allowed values for `Trigger.swImplPolicy` `]()` The **only** allowed values for the attribute `Trigger.swImplPolicy` are either `STANDARD` (in which case the `Trigger` processing does not use a queue) or `QUEUED` (in which case the processing of `Triggers` positively uses a queue). `]()`

For more information regarding the ability to connect different kinds of `PortPrototypes` typed by a `TriggerInterface` to each others please refer to [\[constr_1204\]](#) and [\[constr_1205\]](#).

4.2.5 Communication of Modes

There are two distinctive use cases for the communication of modes via ports:

1. An actual mode transition can be communicated from a mode manager component to its client components to enforce a mode switch.
2. A request for a mode transition can be communicated from any component to a mode manager.

[TPS_SWCT_01087] Propagation of mode information `]()` For communicating a mode switch (i.e. the first use case), the Software-Component Template describes the concept of the communication of `ModeDeclarationGroupPrototypes` similar to the communication of `VariableDataPrototypes` but is uses a special type of `PortInterface`: the collections of `ModeDeclarations` that are required or provided by a `SwComponentType` are defined (as depicted in Figure 4.6) by means of `ModeSwitchInterfaces` used to type the `PortPrototypes` owned by the `SwComponentType`. `](RS_SWCT_03203)`

Due to the strong interaction with the RTE for handling the mode switches, this first use case does not allow communication across ECU boundaries:

[constr_4000] Local communication of mode switches `]()` Ports with `ModeSwitchInterfaces` cannot be connected across ECU boundaries. `]()`

[constr_2049] Different `ModeDeclarationGroups` shall have different `short-Names`. `]()` A software component is not allowed to type multiple `PortPrototypes` with `ModeSwitchInterfaces` where the contained `ModeDeclarationGroupPro-`

`types` are referencing `ModeDeclarationGroups` with identical `shortNames` but different `ModeDeclarations`. `()`

Obviously, the rationale for [constr_2049] is to avoid conflicts in generated RTE files.

For instance:

Two `ModeDeclarationGroups` with identical `shortName` “Foo” are defined.

`ModeDeclarationGroup` “Foo”
contains the `ModeDeclarations` “X”, “Y”, “Z”

`ModeDeclarationGroup` “Foo*”
contains `ModeDeclarations` “W”, “X”, “Y”, “Z”

In this case a software component is only allowed to use either “Foo” or “Foo*”

Class	ModeSwitchInterface			
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
Note	A mode switch interface declares a <code>ModeDeclarationGroupPrototype</code> to be sent and received. Tags: <code>atp.recommendedPackage=PortInterfaces</code>			
Base	ARElement , ARObject , AtpBlueprint , AtpBlueprintable , AtpClassifier , AtpType , CollectableElement , Identifiable , MultilanguageReferrable , PackageableElement , PortInterface , Referrable			
Attribute	Datatype	Mul.	Kind	Note
modeGroup	ModeDeclarationGroupPrototype	1	aggr	The <code>ModeDeclarationGroupPrototype</code> of this mode interface.

Table 4.17: ModeSwitchInterface

Class	ModeDeclarationGroupPrototype			
Package	M2::AUTOSARTemplates::CommonStructure::ModeDeclaration			
Note	The <code>ModeDeclarationGroupPrototype</code> specifies a set of Modes (<code>ModeDeclarationGroup</code>) which is provided or required in the given context.			
Base	ARObject , AtpFeature , AtpPrototype , Identifiable , MultilanguageReferrable , Referrable			
Attribute	Datatype	Mul.	Kind	Note
swCalibrationAccess	SwCalibrationAccessEnum	0..1	attr	This allows for specifying whether or not the enclosing <code>ModeDeclarationGroupPrototype</code> can be measured at run-time.
type	ModeDeclarationGroup	1	trf	The "collection of <code>ModeDeclarations</code> " (= <code>ModeDeclarationGroup</code>) supported by a component Stereotypes: <code>isOfType</code>

Table 4.18: ModeDeclarationGroupPrototype

Please note that by aggregating `SwCalibrationAccessEnum` in the role `swCalibrationAccess` a `ModeDeclarationGroupPrototype` gains the ability to become measurable. This implies the following constraint:

[constr_1172] Allowed values of `SwCalibrationAccessEnum` for `ModeDeclarationGroupPrototype` [The only allowed values of `swCalibrationAccess` aggregated by `ModeDeclarationGroupPrototype` are `notAccessible` and `readOnly`.]()

Enumeration	SwCalibrationAccessEnum
Package	M2::AUTOSARTemplates::CommonStructure::DataDefProperties
Note	Determines the access rights to a data object w.r.t. measurement and calibration.
Literal	Description
notAccessible	The element will not be accessible via MCD tools, i.e. will not appear in the ASAP file.
readOnly	The element will only appear as read-only in an ASAP file.
readWrite	The element will appear in the ASAP file with both read and write access.

Table 4.19: SwCalibrationAccessEnum

[TPS_SWCT_01566] Define literals for an MCD system in the context of a `FlatInstanceDescriptor` [If `ModeDeclarationGroupPrototype.swCalibrationAccess` is set to `readOnly` a referenced `FlatInstanceDescriptor.swDataDefProps` may in turn refer to a `CompuMethod` that defines the particular literals used in the MCD system for displaying values of the the measured `ModeDeclarationGroupPrototypes`.]([RS_SWCT_03203](#))

The existence of this use case is the reason for putting “AI” at the intersection of `compuMethod` and `FlatInstanceDescriptor`.

Another possible scenario (that does not necessarily have to be related to `ModeDeclarationGroupPrototypes` but to the definition of literals for MCD systems in general) is that a `FlatInstanceDescriptor` does not exist (e.g. because the affected piece of data exists in the basic software) but still it would be good to have the ability to define particular literals for displaying values in an MCD system.

This case can be supported by the AUTOSAR standard as well by putting “AI” at the intersection of `compuMethod` and `McDataInstance` in table 5.39.

[TPS_SWCT_01200] `ModeDeclarationGroupPrototype` per `ModeSwitchInterface` [The multiplicity of the aggregation of `ModeDeclarationGroupPrototype` to `ModeSwitchInterface` is pragmatically limited to 1.]([RS_SWCT_03203](#))

Admittedly, there would be no technical restriction to support a 0..* multiplicity but on the other hand it does not seem as if any reasonable use case for such a scenario exists.

If somehow a `SwComponentType` would have to consider two or even more `ModeDeclarationGroupPrototypes` it is very likely that these would be part of different `ModeSwitchInterfaces`.

The containment of a [ModeDeclarationGroupPrototype](#) in a [ModeSwitchInterface](#) allows for explicitly defining [SwConnectors](#) which communicate between [SwComponentPrototypes](#) and to define service interfaces for communication with [ServiceSwComponentTypes](#). Due to the compatibility rules of [PortInterfaces](#) (see chapter 6) each [SwComponentType](#) can rely on the availability of required mode activations.

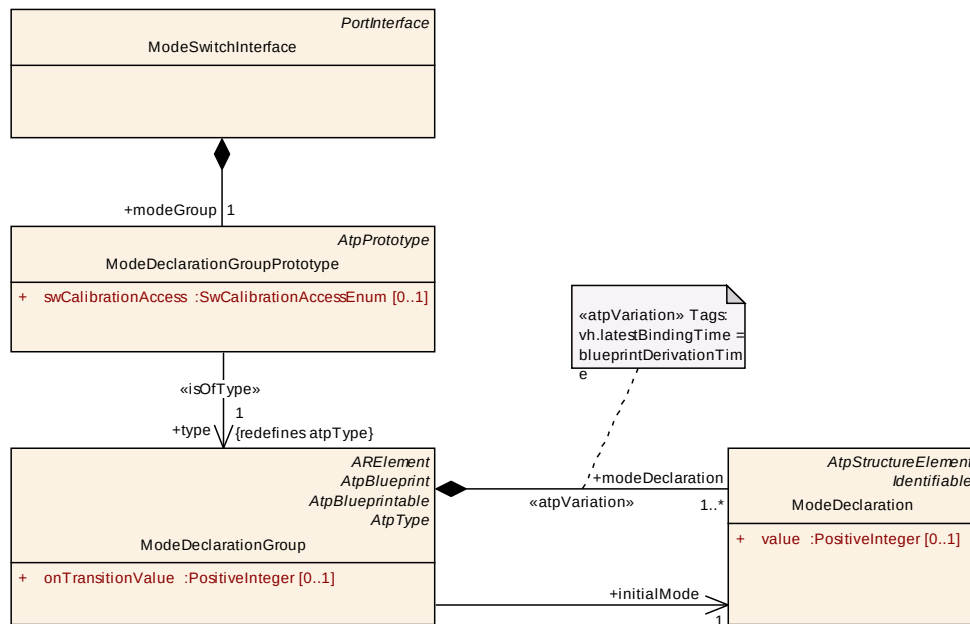


Figure 4.6: Mode Switch Interface

Please note that each [SwComponentType](#) can define (via their [PortPrototypes](#) and [ModeSwitchInterfaces](#)) a list of required and provided [ModeDeclarationGroupPrototypes](#).

[TPS_SWCT_01201] CompositionSwComponentType requires and provides the modes that are required or provided by its contained [SwComponentPrototypes](#). Eventually, a [CompositionSwComponentType](#) requires and provides the modes that are required or provided by its contained [SwComponentPrototypes](#). The delegation of these modes from [SwComponentPrototypes](#) to the enclosing [CompositionSwComponentType](#) is explicitly described by [DelegationSwConnectors](#). [\(RS_SWCT_03202, RS_SWCT_03203\)](#)

The formal description of a software-component does not make any assumptions about the semantics of the required and provided [ModeDeclarationGroupPrototypes](#). It just requires and provides the [ModeDeclarationGroupPrototypes](#) by name. For more information about mode declaration refer to section 9.1.

[TPS_SWCT_01086] Request mode change [The ability to request a mode (i.e. the second use case) is modeled on the VFB via a [SenderReceiverInterface](#) and for the RTE it is like a usual communication, that means the connector can also cross ECU boundaries and the communicated [dataElements](#) have to be based on [AutosarDataTypes](#). [\(RS_SWCT_03202, RS_SWCT_03203\)](#)

However, for semantic consistency with the first use case, a communicated mode request shall also be mapped to a corresponding [ModeDeclarationGroup](#). This can be defined by a mapping class as shown in figure 4.7.

The [ImplementationDataType](#) mapped to a certain [ModeDeclarationGroup](#) can then be used in a [PortInterface](#) to represent a [ModeDeclaration](#) of the associated [ModeDeclarationGroup](#) as a numerical value:

[constr_4002] Unambiguous mapping of modes to data types [Within one [DataTypeMappingSet](#), a [ModeDeclarationGroup](#) shall not be mapped to different [ImplementationDataTypes](#).]()

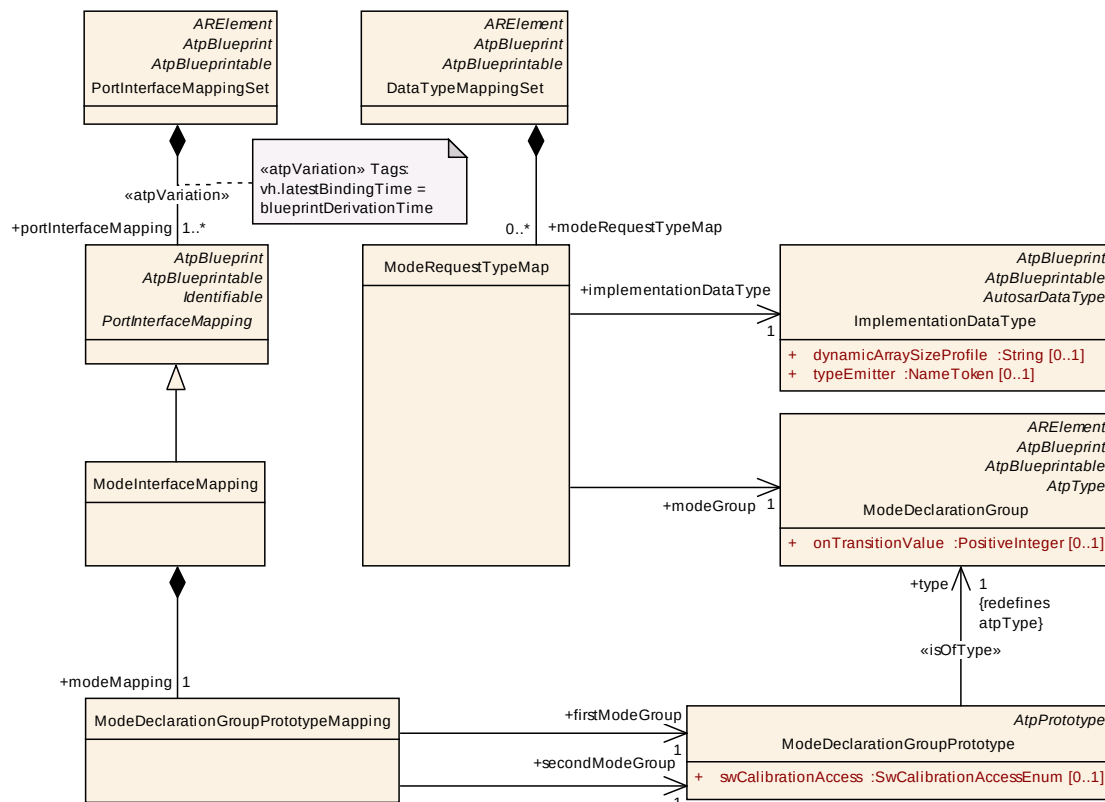


Figure 4.7: Mapping of modes to data types

Class	ModeRequestTypeMap				
Package	M2::AUTOSARTemplates::CommonStructure::ModeDeclaration				
Note	Specifies a mapping between a ModeDeclarationGroup and an ImplementationDataType. This ImplementationDataType shall be used to implement the ModeDeclarationGroup.				
Base	ARObject				
Attribute	Datatype	Mul.	Kind	Note	
implementationDataType	ImplementationDataType	1	ref	This is the corresponding ImplementationDataType. It shall be modeled along the idea of an "unsigned integer-like" data type.	
modeGroup	ModeDeclarationGroup	1	ref	This is the corresponding ModeDeclarationGroup.	

Attribute	Datatype	Mul.	Kind	Note

Table 4.20: ModeRequestTypeMap

[constr_1166] Restrictions of ModeRequestTypeMap [For every ModeDeclarationGroup referenced by a ModeDeclarationGroupPrototype used in a PortPrototype typed by a ModeSwitchInterface a ModeRequestTypeMap shall exist that points to the ModeDeclarationGroup and also to an eligible ImplementationDataType.

The ModeRequestTypeMap shall be aggregated by a DataTypeMappingSet which is referenced from the SwcInternalBehavior that is owned by the ApplicationSwComponentType that also owns the PortPrototype.]()

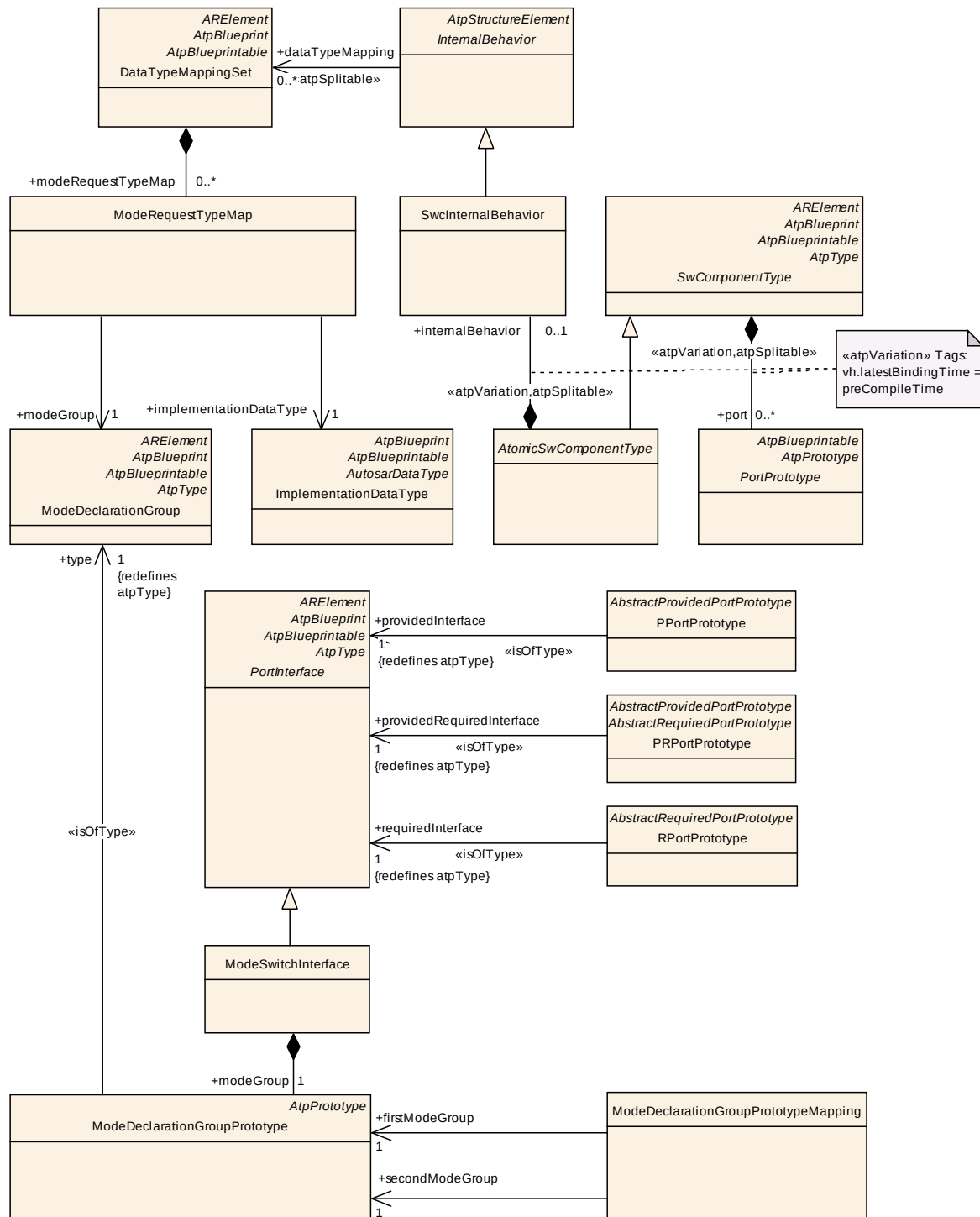


Figure 4.8: Big picture of mode declaration mapping

[constr_1167] **ImplementationDataTypes** used as **ModeRequestTypeMap.implementationDataType** [The **ImplementationDataType** referenced by a **ModeRequestTypeMap** shall **either** be of category **VALUE** or of category **TYPE_REFERENCE** that in turn references an **ImplementationDataType** of category **VALUE**.

The `baseType` referenced by the `ImplementationDataType` shall have set the value of the attribute `BaseTypeDirectDefinition.baseTypeEncoding` to `NONE`.
 ⌋()

[TPS_SWCT_01202] `ApplicationDataType` defines a subset of the values used in the `ModeDeclarationGroup` ⌈ Please note that the corresponding `ApplicationDataType` is defining a subset of the values used in the `ModeDeclarationGroup` and the used labels may differ from the names used for the `ModeDeclarations`.

It is in the responsibility of a system designer to maintain the data types and `ModeDeclarationGroups` according to the functional needs.

For example, a `ModeRequester` may only request a subset of the available Modes (via `SenderReceiverInterface` or `ClientServerInterface`). The `ModeManager` may additionally decide to indicate failure. ⌋(*RS_SWCT_03203*)

For more information regarding the ability to connect different kinds of `PortPrototypes` typed by a `ModeSwitchInterface` to each other please refer to [*constr_1204*] and [*constr_1205*].

4.2.6 Parameter Communication

Of course, the “communication” of `ParameterDataPrototypes` as part of a `ParameterInterface` does not establish an actual transmission of data.

The term is used in a conceptual meaning; and the existence of something like a `ParameterInterface` is justified by the mere idea of unifying the exposure of calibration parameters at the surface of a software-component on the same formal level as the exposure of other pieces of data, i.e. by means of a `PortPrototype` typed by a `PortInterface`.

[constr_1312] `PortPrototypes` typed by a `ParameterInterface` ⌈ `PortPrototypes` typed by a `ParameterInterface` can either be `PPortPrototypes` or `RPortPrototypes`. The usage of `PRPortPrototypes` that are typed by a `ParameterInterface` is not supported. ⌋()

4.3 PortInterface Mapping and Data Scaling

In former versions of this specification, the requirements on `PortInterfaces` to match each other could lead to situations where `PortInterfaces` that were “practically” compatible would nevertheless be rejected because of formal reasons (e.g. `ShortNames` of `dataElements` do not match).

In order to also support scenarios where the developer of a `CompositionSwComponentType` needs to connect `PortPrototypes` that would match to each others

but don't fulfill formal requirements the concept of "port interface mapping" has been introduced.

[TPS_SWCT_01158] Three cases for `PortInterfaceMapping` [In general there are three different cases, where a `PortInterfaceMapping` is suitable.

1. Two `PortPrototypes` shall be connected and the `PortInterface` elements are compatible except the unequal `shortNames`. This requires a pure logical mapping of the `PortInterface` elements.
2. `PortInterface` elements are logically equivalent but the range and resolution is differently. This requires a data conversion respectively a re-scaling of the provided data and arguments to the required data and arguments range and resolution.
3. `invalidationPolicy` of `PortInterface` elements is different. This might require the implementation of different invalidation handling strategies for the same `dataElement` in parallel on the same ECU.
4. Two `PortPrototypes` shall be connected and the `PortInterface` elements shall be converted using the AUTOSAR data transformer approach.

]([RS_SWCT_03210](#))

Typically the mapping of such `PortInterface` is agreed once between the different component vendors and system designer in the early phase of a project.

One (prominent) use-case for item 4 in [\[TPS_SWCT_01158\]](#) is the interaction between the `NvBlockSwComponentType` and the AUTOSAR Dcm.

Specifically, the RTE will call a data transformer to convert the `uint8`-array representation of the diagnostic data available from a `PortPrototype` owned by the Dcm `ServiceSwComponentType` to a `VariableDataPrototype` owned by a `PortPrototype` of `NvBlockSwComponentType`.

For the configuration of this purpose, the applicable `DataPrototypeMapping` refers to a `DataTransformation` in the role `firstToSecondDataTransformation` (see Figure 4.9).

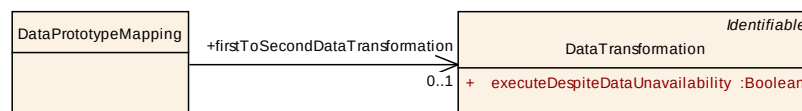


Figure 4.9: Configuration of Ecu-internal data transformation

Note that for this specific interaction between an `ApplicationSwComponentType` and a `ServiceSwComponentType` [\[TPS_SWCT_01579\]](#) applies which defines that attribute `isService` shall be set to false for the `dataElements` in `PortPrototypes` typed by a `SenderReceiverInterface`.

[TPS_SWCT_01159] Mapping is described separately from the `SwConnector` as reusable `ARElement` [The mapping is described separately from the `SwConnec-`

tor as reusable `ARElement`. A set of `PortInterfaceMappings` is grouped in a `PortInterfaceMappingSet`. [\]\(RS_SWCT_03210\)](#)

[TPS_SWCT_01543] `PortInterfaceMapping` overrides all other compatibility rules [\[](#) The existence of a `PortInterfaceMapping` overrides all other compatibility rules given that the following statements are fulfilled:

- [\[constr_1071\]](#) applies also for the application of a `PortInterfaceMapping`.
- [\[constr_1268\]](#) applies also for the application of a `PortInterfaceMapping`.
- [\[constr_1269\]](#) applies also for the application of a `PortInterfaceMapping`.
- [\[constr_1270\]](#) applies also for the application of a `PortInterfaceMapping`.
- A structural difference between mapped `DataPrototypes` can be mitigated by means of a `SubElementMapping`. This includes the case that a “structure” data type is mapped to an “array” data type and vice versa. [\[TPS_SWCT_01195\]](#) is also applicable.

When using a `PortInterfaceMapping`, the developer of a software-component needs to properly understand the consequences in terms of model semantics. [\]\(RS_SWCT_03210\)](#)

Please note that [\[TPS_SWCT_01543\]](#) does not require a tool implementation to ignore and let go unreported deviations of all other compatibility rules in the presence of a `PortInterfaceMapping`.

If this is considered helpful, the tool **may** still issue warnings with respect to compatibility rules defined in section 6 but this is not mandated by the AUTOSAR standard. The tool, however, **shall not** report errors in this case.

Class	PortInterfaceMappingSet			
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
Note	Specifies a set of (one or more) <code>PortInterfaceMappings</code> . Tags: atp.recommendedPackage=PortInterfaceMappingSets			
Base	ARElement , ARObject , AtpBlueprint , AtpBlueprintable , CollectableElement , Identifiable , MultilanguageReferrable , PackageableElement , Referrable			
Attribute	Datatype	Mul.	Kind	Note
portInterfaceMapping	PortInterfaceMapping	1..*	aggr	Specifies one <code>PortInterfaceMapping</code> to support the connection of Ports typed by two different <code>PortInterfaces</code> with <code>PortInterface</code> elements having unequal names and/or unequal semantic (resolution or range). Stereotypes: atpVariation Tags: vh.latestBindingTime=blueprintDerivationTime

Table 4.21: PortInterfaceMappingSet

Class	PortInterfaceMapping (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
Note	Specifies one PortInterfaceMapping to support the connection of Ports typed by two different PortInterfaces with PortInterface elements having unequal names and/or unequal semantic (resolution or range).			
Base	ARObject, AtpBlueprint , AtpBlueprintable, Identifiable , Multilanguage Referrable, Referrable			
Attribute	Datatype	Mul.	Kind	Note
–	–	–	–	–

Table 4.22: PortInterfaceMapping

4.3.1 PortInterface Mapping

By default, the [shortNames](#) of [PortInterface](#) elements are used to identify the matching element pairs of connected [PortPrototypes](#). In case of non-matching [shortNames](#) (this might be due to distributed development, off-the-shelves development, or reuse of software-components) it is required to explicitly specify which elements of [PortInterfaces](#) shall correlate to each other.

This definition is provided with [PortInterfaceMappings](#).

[TPS_SWCT_01099] PortInterfaceMapping [Each [PortInterfaceMapping](#) describes the mapping of the [PortInterface](#) elements of exactly two [PortInterfaces](#).] ([RS_SWCT_03155](#), [RS_SWCT_03210](#))

To apply the [PortInterfaceMapping](#) a [SwConnector](#) has to reference a [PortInterfaceMapping](#).

[constr_1151] Applicability of PortInterfaceMapping [A [PortInterfaceMapping](#) is only applicable and valid for a [SwConnector](#) if the two [PortPrototypes](#) which are referenced by the [SwConnector](#) are typed by the same two [PortInterfaces](#) which are mapped by the [PortInterfaceMapping](#).]()

[TPS_SWCT_01100] Precedence of PortInterfaceMapping [The mapping via [PortInterfaceMapping](#) has a higher precedence than the mapping by equal [shortNames](#) as defined in chapter 6. If a connector has an associated [PortInterfaceMapping](#) this mapping shall be strictly binding with respect to the number of mapped data elements.] ([RS_SWCT_03155](#), [RS_SWCT_03210](#))

[TPS_SWCT_01101] Unmapped elements of PortInterfaces [Unmapped [PortInterface](#) elements will not be connected by the referencing [SwConnector](#).] ([RS_SWCT_03155](#), [RS_SWCT_03210](#))

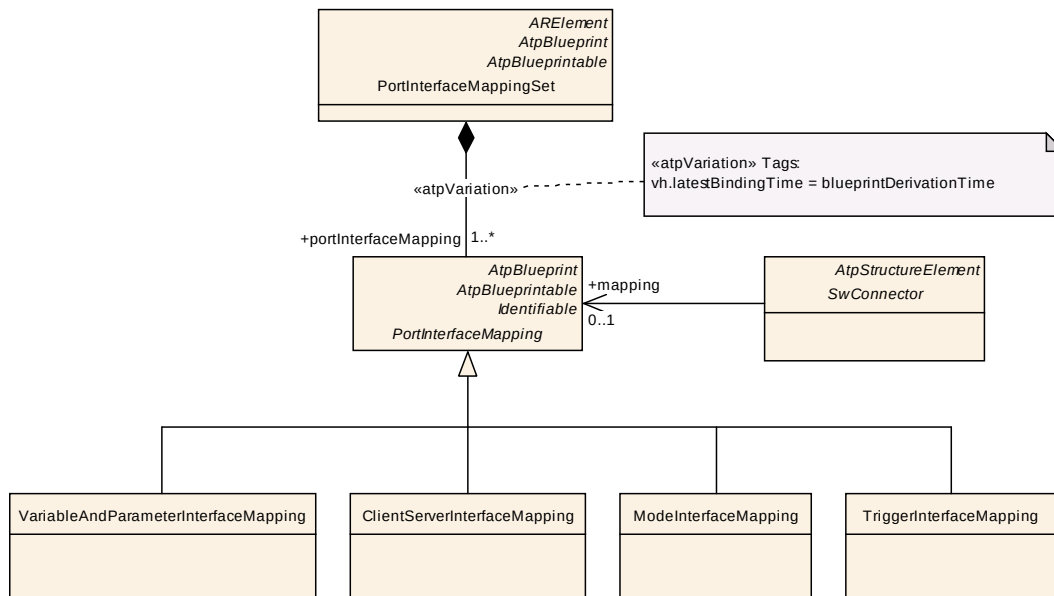


Figure 4.10: Relevant meta-classes for PortInterface element mapping

4.3.1.1 Mapping of Sender Receiver Interface, Parameter Interface and Non Volatile Data Interface Elements

[TPS_SWCT_01102] **VariableAndParameterInterfaceMapping** [The `VariableAndParameterInterfaceMapping` defines the correlation of `VariableDataPrototypes` and `ParameterDataPrototypes` defined in the context of `DataInterfaces`, i.e. `SenderReceiverInterface`, `NvDataInterface`, or `ParameterInterface`.]([RS_SWCT_03155](#), [RS_SWCT_03210](#), [RS_SWCT_03170](#))

[constr_1159] **Consistency of VariableAndParameterInterfaceMapping with respect to the referenced DataInterfaces** [Within one `VariableAndParameterInterfaceMapping` all `firstDataPrototypes` shall belong to one and only one `DataInterface` and all `secondDataPrototypes` shall belong to one other and only one other `DataInterface`.]()

[TPS_SWCT_01103] **Mapping between different kinds of PortInterfaces** [Thereby it is possible to describe the mapping between different kinds of `PortInterfaces` for instance a `ParameterInterface` and `SenderReceiverInterface`.]([RS_SWCT_03155](#), [RS_SWCT_03210](#), [RS_SWCT_03170](#))

[TPS_SWCT_01104] **Possible mappings are restricted by the swImplPolicy** [Nevertheless, the possible mappings of `VariableDataPrototypes` and `ParameterDataPrototypes` are restricted by the `swImplPolicy` attribute.]([RS_SWCT_03155](#), [RS_SWCT_03210](#), [RS_SWCT_03170](#))

[constr_1039] **Relevance of swImplPolicy** [It is not possible to define a mapping between an element where the `swImplPolicy` is set to `queued` and an other element where the `swImplPolicy` is set differently.]()

This is required to fulfill the compatibility rules defined in table 6.1

[constr_1040] Conversion of `SenderReceiverInterfaces` [The conversion of elements of `SenderReceiverInterfaces` is possible if one of the following conditions applies:

- The `AutosarDataTypes` of the referred `DataPrototypes` are compatible as described in chapter 6.2.
- A conversion of the data as described in chapter 4.3.2 is available.
- A `DataPrototypeMapping.firstToSecondDataTransformation` is defined.

]()

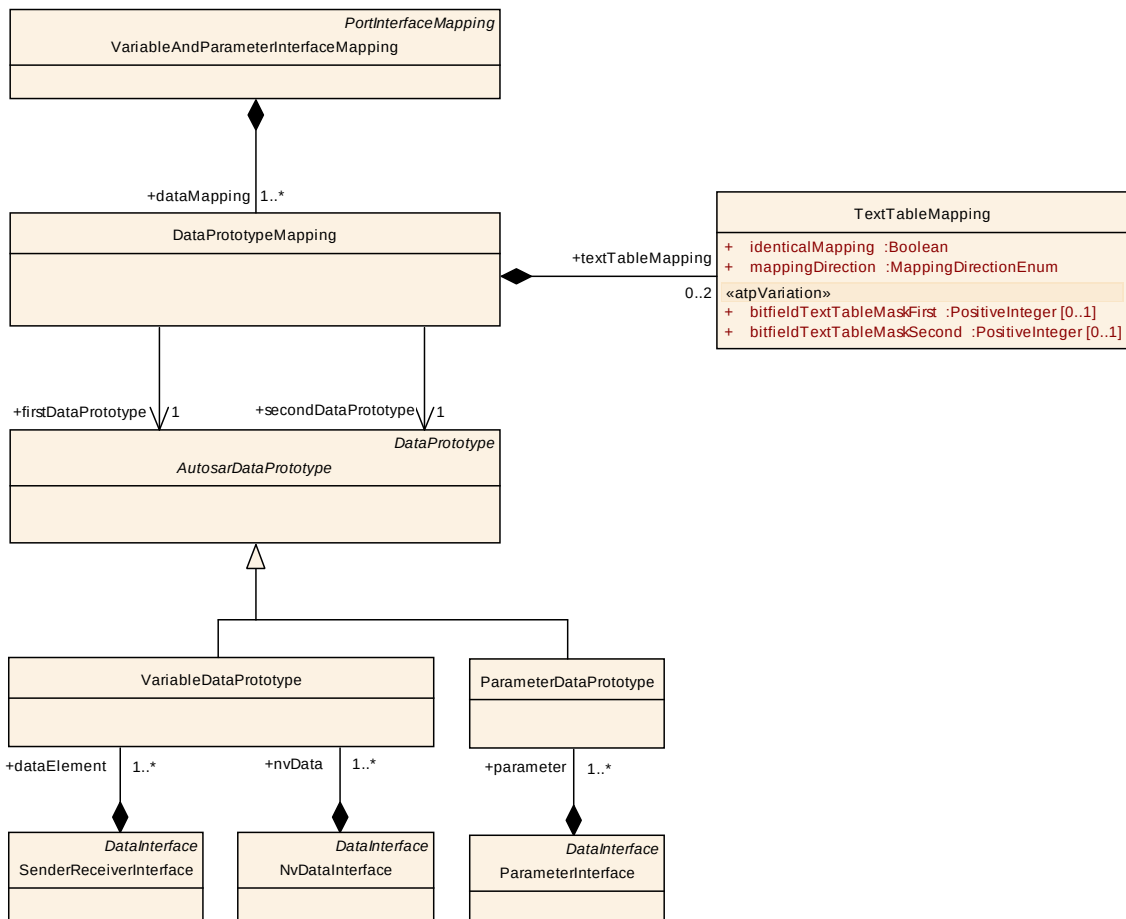


Figure 4.11: Mapping of Sender Receiver Interface, Parameter Interface and Non Volatile Data Interface elements

Class	VariableAndParameterInterfaceMapping			
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
Note	Defines the mapping of VariableDataPrototypes or ParameterDataPrototypes in context of two different SenderReceiverInterfaces, NvDataInterfaces or ParameterInterfaces.			
Base	ARObject, AtpBlueprint , AtpBlueprintable, Identifiable , MultilanguageReferrable, PortInterfaceMapping , Referrable			
Attribute	Datatype	Mul.	Kind	Note
dataMapping	DataPrototypeMapping	1..*	aggr	Defines the mapping of two particular VariableDataPrototypes or ParameterDataPrototypes with unequal names and/or unequal semantic (resolution or range) in context of two different SenderReceiverInterfaces, NvDataInterfaces or ParameterInterfaces

Table 4.23: VariableAndParameterInterfaceMapping

Class	DataPrototypeMapping			
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
Note	Defines the mapping of two particular VariableDataPrototypes, ParameterDataPrototypes or ArgumentDataPrototypes with unequal names and/or unequal semantic (resolution or range) in context of two different SenderReceiverInterface, NvDataInterface or ParameterInterface or Operations. If the semantic is unequal following rules apply: The textTableMapping is only applicable if the referred DataPrototypes are typed by AutosarDataType referring to CompuMethods of category TEXTTABLE, SCALE_LINEAR_AND_TEXTTABLE or BITFIELD_TEXTTABLE. In the case that the DataPrototypes are typed by AutosarDataType either referring to CompuMethods of category LINEAR, IDENTICAL or referring to no CompuMethod (which is similar as IDENTICAL) the linear conversion factor is calculated out of the factorSiToUnit and offsetSiToUnit attributes of the referred Units and the CompuRationalCoeffs of a compuInternalToPhys of the referred CompuMethods.			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
firstDataPrototype	AutosarDataPrototype	1	ref	First to be mapped DataPrototype in context of a SenderReceiverInterface, NvDataInterface, ParameterInterface or Operation.
firstToSecondDataTransformation	DataTransformation	0..1	ref	This defines the need to execute the DataTransformation <Mip>_<transformerId> functions of the transformation chain when communicating from the DataPrototypeMapping.firstDataPrototype to the DataPrototypeMapping.secondDataPrototype. And to execute the DataTransformation <Mip>_Inv_<transformerId> functions of the transformation chain when communicating from the DataPrototypeMapping.secondDataPrototype to the DataPrototypeMapping.firstDataPrototype.
secondDataPrototype	AutosarDataPrototype	1	ref	Second to be mapped DataPrototype in context of a SenderReceiverInterface, NvDataInterface, ParameterInterface or Operation.

Attribute	Datatype	Mul.	Kind	Note
subElementMapping	SubElementMapping	*	aggr	This represents the owned SubelementMapping.
textTableMapping	TextTableMapping	0..2	aggr	Applied TextTableMapping(s)

Table 4.24: DataPrototypeMapping

4.3.1.2 Mapping of Client Server Interface Elements

[TPS_SWCT_01105] **ClientServerInterfaceMapping** [The `ClientServerInterfaceMapping` defines the correlation of `ClientServerOperations` defined in the context of `ClientServerInterfaces`.](*RS_SWCT_03155*, *RS_SWCT_03210*)

[constr_1041] **Conversion of ClientServerInterfaces** [Either the `AutosarDataTypes` of the referred `ArgumentDataPrototypes` are compatible as described in chapter 6.2 or a conversion of the data as described in chapter 4.3.2 is available.]()

[constr_1237] **Scope of mapped ClientServerOperations in the context of a ClientServerOperationMapping** [All `ClientServerOperations` referenced by a `ClientServerOperationMapping` in the role `firstOperation` shall belong to exactly one `ClientServerInterface`.

All `ClientServerOperations` referenced by a `ClientServerOperationMapping` in the role `secondOperation` shall belong to exactly one other `ClientServerInterface`.]()

[constr_1238] **Scope of mapped ApplicationErrors in the context of a ClientServerOperationMapping** [All `ApplicationErrors` referenced by a `ClientServerApplicationErrorMapping` in the role `firstApplicationError` shall belong to exactly one `ClientServerInterface`.

All `ApplicationErrors` referenced by a `ClientServerApplicationErrorMapping` in the role `secondApplicationError` shall belong to exactly one other `ClientServerInterface`.]()

[constr_1240] **Consistency of ArgumentDataPrototypes within the context of a ClientServerOperationMapping** [For each `argument` owned by a `ClientServerOperationMapping.firstOperation` and `ClientServerOperationMapping.secondOperation` a reference in the role `ClientServerOperationMapping.argumentMapping.firstDataPrototype` or `ClientServerOperationMapping.argumentMapping.secondDataPrototype` shall exist originated by one of the `ClientServerOperationMapping.argumentMappings` owned by the mentioned `ClientServerOperationMapping`.]()

[constr_1268] **ArgumentDataPrototype.direction shall be preserved in a ClientServerOperationMapping** [Within the context of a `ClientServerOperationMapping`

ationMapping, the value of the argument `ArgumentDataPrototype.direction` of two mapped `ArgumentDataPrototype` shall be identical. $\square()$

[constr_1269] Number of arguments shall be preserved in a ClientServerOperationMapping $\square$ Within the context of a `ClientServerOperationMapping`, the number of arguments of `firstOperation` and `secondOperation` shall be identical. $\square()$

[constr_1270] ArgumentDataPrototype shall be mapped only once in a ClientServerOperationMapping $\square$ Within the context of a `ClientServerOperationMapping`, each argument shall only be referenced **once** in the role `firstDataPrototype` or `secondDataPrototype`. $\square()$

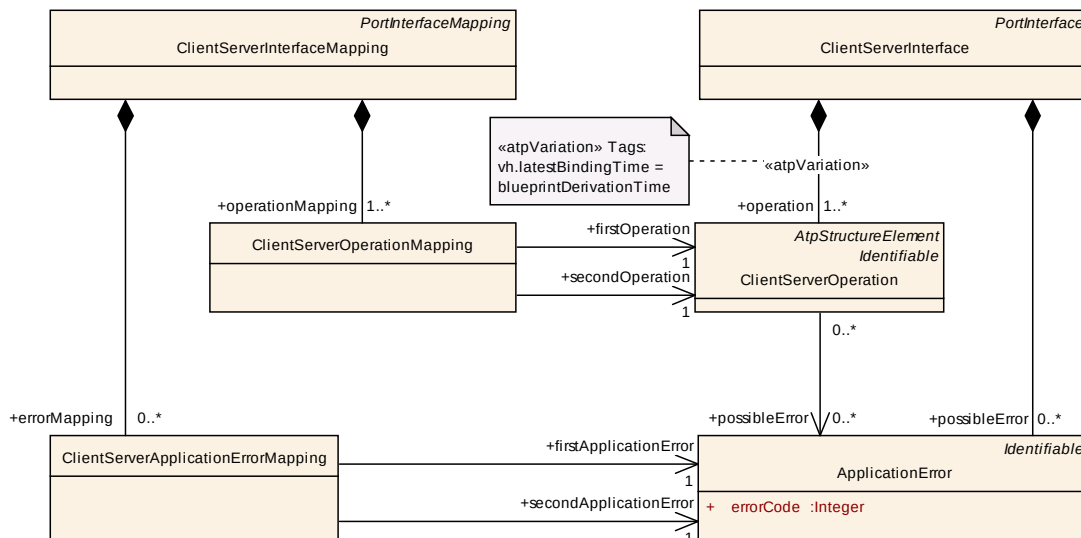


Figure 4.12: Mapping of `ClientServerInterface` elements and mapping of arguments

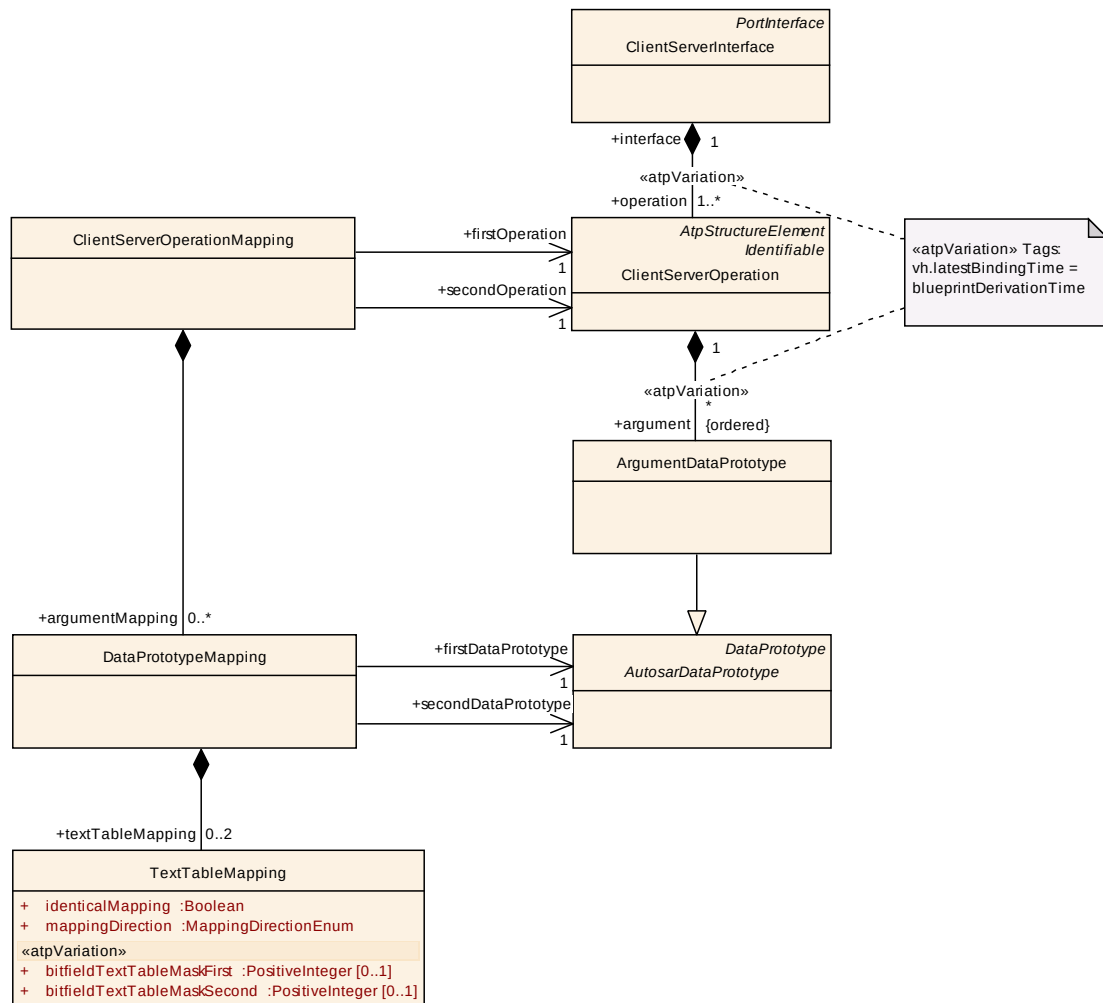


Figure 4.13: Mapping of **ArgumentDataPrototypes**

Class	ClientServerInterfaceMapping			
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
Note	Defines the mapping of ClientServerOperations in context of two different ClientServerInterfaces.			
Base	ARObject, AtpBlueprint , AtpBlueprintable , Identifiable , MultilanguageReferrable , PortInterfaceMapping , Referrable			
Attribute	Datatype	Mul.	Kind	Note
errorMapping	ClientServerApplicationErrorMapping	*	aggr	Map two different ApplicationErrors defined in the context of two different ClientServerInterfaces.
operationMapping	ClientServerOperationMapping	1..*	aggr	Mapping of two ClientServerOperations in two different ClientServerInterfaces

Table 4.25: ClientServerInterfaceMapping

Class	ClientServerOperationMapping			
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
Note	Defines the mapping of two particular ClientServerOperations in context of two different ClientServerInterfaces.			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
argument Mapping	DataPrototypeMapping	*	aggr	Defines the mapping of two particular ArgumentDataPrototypes with unequal names or unequal semantic (resolution or range) in context of Operations.
firstOperation	ClientServerOperation	1	ref	First to-be-mapped ClientServerOperation of a ClientServerInterface.
secondOperation	ClientServerOperation	1	ref	Second to-be-mapped ClientServerOperation of a ClientServerInterface.

Table 4.26: ClientServerOperationMapping

Class	ClientServerApplicationErrorMapping			
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
Note	This meta-class represents the ability to map ApplicationErrors onto each other.			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
firstApplicationError	ApplicationError	1	ref	This represents the first ApplicationError in the context of the ClientServerApplicationErrorMapping.
secondApplicationError	ApplicationError	1	ref	This represents the second ApplicationError in the context of the ClientServerApplicationErrorMapping.

Table 4.27: ClientServerApplicationErrorMapping

4.3.1.3 Mapping of Mode Interface Elements

[TPS_SWCT_01160] [ModeInterfaceMapping](#) [The [ModeInterfaceMapping](#) defines the correlation of [ModeDeclarationGroupPrototypes](#) defined in the context of [ModeSwitchInterfaces](#).]([RS_SWCT_03210](#))

[TPS_SWCT_01167] **Validity of [ModeInterfaceMapping](#)** [The mapping of [ModeDeclarationGroupPrototypes](#) is only valid if these are typed by (read “refer to”) compatible [ModeDeclarationGroups](#) according chapter 6.7.]([RS_SWCT_03210](#))

Class	ModelInterfaceMapping			
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
Note	Defines the mapping of ModeDeclarationGroupPrototypes in context of two different ModelInterfaces.			
Base	ARObject, AtpBlueprint , AtpBlueprintable , Identifiable , MultilanguageReferrable , PortInterfaceMapping , Referrable			
Attribute	Datatype	Mul.	Kind	Note

Attribute	Datatype	Mul.	Kind	Note
modeMapping	ModeDeclarationGroupPrototypeMapping	1	aggr	Mapping of two ModeDeclarationGroupPrototypes in two different ModelInterfaces

Table 4.28: ModelInterfaceMapping

Class	ModeDeclarationGroupPrototypeMapping			
Package	M2::AUTOSARTemplates::CommonStructure::ModeDeclaration			
Note	Defines the mapping of two particular ModeDeclarationGroupPrototypes (in the given context) that are unequally named and/or require a reference to a ModeDeclarationMappingSet in order to become compatible by definition of ModeDeclarationMappings.			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
firstModeGroup	ModeDeclarationGroupPrototype	1	ref	ModeDeclarationGroupPrototype to be mapped.
modeDeclarationMappingSet	ModeDeclarationMappingSet	0..1	ref	This represents the available mappings of ModeDeclarations in the context of this ModeDeclarationGroupPrototype.
secondModeGroup	ModeDeclarationGroupPrototype	1	ref	ModeDeclarationGroupPrototype to be mapped.

Table 4.29: ModeDeclarationGroupPrototypeMapping

[TPS_SWCT_01449] Semantics of a [ModeDeclarationGroupPrototypeMapping](#) | A [ModeDeclarationGroupPrototypeMapping](#) shall be used to identify two [ModeDeclarationGroups](#) that afterwards shall be considered compatible. This also applies if the two [ModeDeclarationGroups](#) deviate with respect to the contained [modeTransitions](#). |(RS_SWCT_03210)

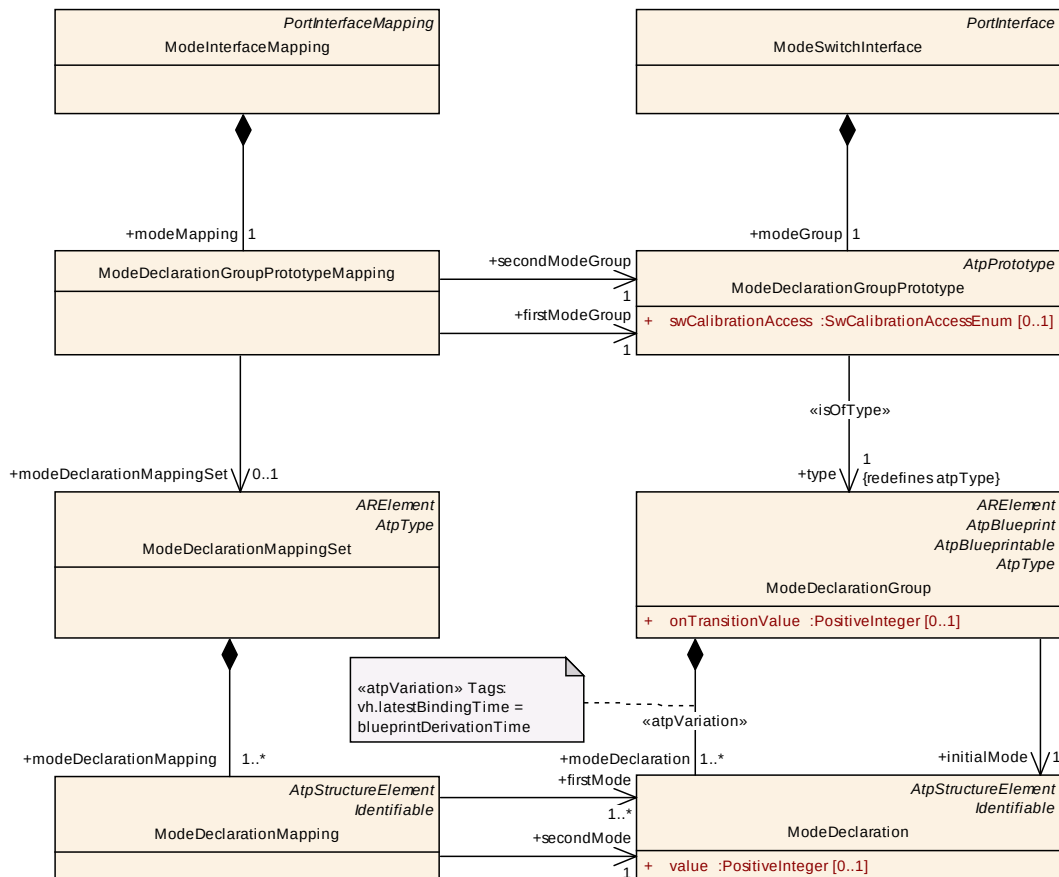


Figure 4.14: Mapping of `ModeSwitchInterface` elements

[constr_1246] Consistency of `firstMode` and `secondMode` in the scope of one `ModeDeclarationMappingSet` | Within the scope of one `ModeDeclarationMappingSet`, all `firstModes` shall belong to one and only one `ModeDeclarationGroup` and all `secondModes` shall belong to one and only one **other** `ModeDeclarationGroup` |()

[constr_1247] Consistency of ModeDeclarationMappingSet with respect to the referenced firstModeGroup and secondModeGroup | If a ModeDeclarationGroupPrototypeMapping.modeDeclarationMappingSet exists, the ModeDeclarationGroup owning the modeDeclarations referenced in the role firstMode shall be the type of the ModeDeclarationGroupPrototypeMapping.firstModeGroup and the ModeDeclarationGroup owning the modeDeclarations referenced in the role secondMode shall be the type of the ModeDeclarationGroupPrototypeMapping.secondModeGroup. |()

[TPS_SWCT_01462] **ModeDeclarationMapping** defines the explicit correlation of **ModeDeclarations** | The meta-class **ModeDeclarationMapping** defines the explicit correlation of **ModeDeclarations** defined in the context of two **ModeDeclarationGroups**. |()

[TPS_SWCT_01463] **modeDeclarationMapping** defines the applicable set of **ModeDeclarationMappings** [The **modeDeclarationMapping** defines the ap-

pllicable set of [ModeDeclarationMappings](#) for the connection of [ModeDeclarationGroupPrototypes](#) typed by [ModeDeclarationGroups](#) with differently named [ModeDeclarations](#) and/or with a different number of [ModeDeclarations](#). $\rfloor()$

Class	ModeDeclarationMappingSet			
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
Note	This meta-class implements a container for ModeDeclarationGroupMappings Tags: atp.recommendedPackage=PortInterfaceMappingSets			
Base	ARElement , ARObject , AtpClassifier , AtpType , CollectableElement , Identifiable , MultilanguageReferrable , PackageableElement , Referrable			
Attribute	Datatype	Mul.	Kind	Note
modeDeclarationMapping	ModeDeclarationMapping	1..*	aggr	This represents the collection of ModeDeclarationMappings owned by the enclosing ModeDeclarationMappingSet.

Table 4.30: ModeDeclarationMappingSet

Class	ModeDeclarationMapping			
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
Note	This meta-class implements a concrete mapping of two ModeDeclarations.			
Base	ARObject , AtpClassifier , AtpFeature , AtpStructureElement , Identifiable , MultilanguageReferrable , Referrable			
Attribute	Datatype	Mul.	Kind	Note
firstMode	ModeDeclaration	1..*	ref	This represents the first ModeDeclaration of the ModeDeclarationMapping. This reference has the multiplicity 1 .. * to support use cases where e.g. one mode of the mode user is mapped to several modes of the mode manager.
secondMode	ModeDeclaration	1	ref	This represents the second ModeDeclaration of the ModeDeclarationMapping.

Table 4.31: ModeDeclarationMapping

[TPS_SWCT_01464] **ModeDeclaration of a mode user is mapped to exactly one ModeDeclaration of a mode manager** $\rfloor$ The mode that corresponds to the [ModeDeclaration](#) of the Mode User is entered or exited when the mode of the mode manager that corresponds to the mapped (i.e. referenced by the same [ModeDeclarationMapping](#)) [ModeDeclaration](#) of the mode manager is entered or exited. $\rfloor$ ([RS_SWCT_03115](#))

[TPS_SWCT_01465] **ModeDeclaration of a mode user is mapped to several ModeDeclarations of a mode manager** $\rfloor$ The mode that corresponds to the mapped [ModeDeclaration](#) of the mode user is entered when any of the modes of the Mode Manager that correspond to [ModeDeclarations](#) referenced by the applicable [ModeDeclarationMapping](#) is entered.

The mode that corresponds to the mapped [ModeDeclaration](#) of the mode user is exited when any of the modes of the Mode Manager that correspond to [ModeDec-](#)

larations referenced by the applicable `ModeDeclarationMapping` is exited if the new mode is not mapped to related mode of the mode user. [\]\(RS_SWCT_03115\)](#)

Please note if one `ModeDeclaration` of a mode user is mapped to **several** `ModeDeclarations` of a mode manager by means of several `ModeDeclarationMappings` the intended semantics is defined in a way that the individual mode transitions of the mode manager are representing “exit” and “enter” events for the Mode User. In other words, the individual transitions are recognizable by the mode user.

If one `ModeDeclaration` of a mode user is (by utilizing the multiplicity of the role `firstMode`) mapped to several `ModeDeclarations` of a mode manager in the context of a single `ModeDeclarationMapping` the semantics is defined in a way that the individual mode transitions of the Mode Manager are not recognizable to the Mode User.

[constr_1209] Mapping of `ModeDeclarations` of mode user to `ModeDeclaration` of mode manager [\[](#) A configuration that maps **several** `ModeDeclarations` representing modes of a mode user to **one** `ModeDeclaration` representing a mode of a mode manager shall be rejected. [\]\(](#)

[constr_1210] Mapping of `ModeDeclarations` of mode user to all `ModeDeclarations` of mode manager [\[](#) If a `ModeDeclarationMapping` exists that references a `ModeDeclaration` representing a mode of the mode manager then `ModeDeclarationMappings` shall exist that map all modes of the mode manager to modes of the mode user. [\]\(](#)

Please note that [\[constr_1210\]](#) prevents the existence of configurations where the mode user is not in a defined mode when no transition is ongoing.

[TPS_SWCT_01545] `ModeDeclaration` of a *mode user* that is not mapped to a `ModeDeclaration` of a *mode manager* [\[](#) A `ModeDeclaration` of a *mode user* that is not mapped to a `ModeDeclaration` of a *mode manager* represents a valid model. In this case the related mode is never entered nor exit during runtime of the ECU. [\]\(RS_SWCT_03115\)](#)

4.3.1.4 Mapping of Trigger Interface Elements

[TPS_SWCT_01161] `TriggerInterfaceMapping` [\[](#) The `TriggerInterfaceMapping` defines the correlation of `Triggers` defined in the context `TriggerInterfaces`. [\]\(RS_SWCT_03210\)](#)

Class	TriggerInterfaceMapping			
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
Note	Defines the mapping of unequal named Triggers in context of two different TriggerInterfaces.			
Base	ARObject, AtpBlueprint , AtpBlueprintable, Identifiable , MultilanguageReferrable, PortInterfaceMapping , Referrable			
Attribute	Datatype	Mul.	Kind	Note
triggerMapping	TriggerMapping	1..*	aggr	Mapping of two Trigger in two different TriggerInterface

Table 4.32: TriggerInterfaceMapping

Class	TriggerMapping			
Package	M2::AUTOSARTemplates::CommonStructure::TriggerDeclaration			
Note	Defines the mapping of two particular unequally named Triggers in the given context.			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
firstTrigger	Trigger	1	ref	A Trigger to be mapped.
secondTrigger	Trigger	1	ref	A Trigger to be mapped.

Table 4.33: TriggerMapping

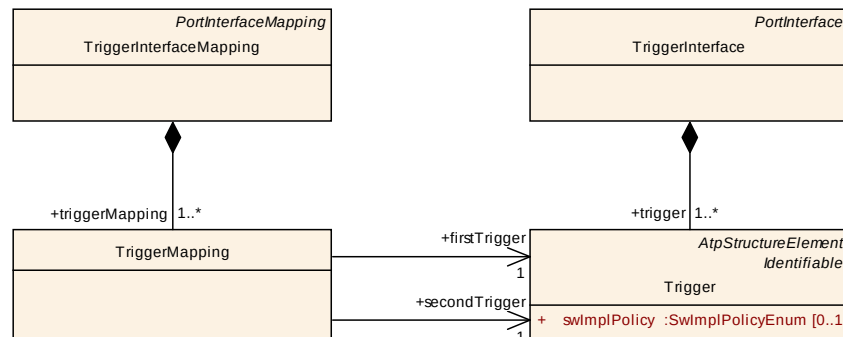


Figure 4.15: Mapping of [TriggerInterface](#) elements

4.3.1.5 Mapping of Elements of a composite Data Type

The mapping of elements of [PortInterfaces](#) is not limited to mapping entire [DataPrototypes](#) onto each others.

[TPS_SWCT_01023] Mapping of elements of composite data types [For applications of [DataInterfaces](#) it is also possible to formally describe the mapping of elements of [ApplicationCompositeDataTypes](#) or [ImplementationDataTypes](#) of category [STRUCTURE](#) or [ARRAY](#) onto each others.] ([RS_SWCT_03210](#), [RS_SWCT_03135](#))

This ability can be used if e.g. `dataElements` on the sender and receiver side are typed by different `ApplicationRecordDataTypes`.

In this case the mapping of elements of `ApplicationCompositeDataTypes` or `ImplementationDataTypes` of category `STRUCTURE` or `ARRAY` onto each others allows for the definition of specific pairs of elements that fulfill the compatibility rules.

[TPS_SWCT_01551] Mapping of elements on the sender side to elements on the receiver side [Unless the attribute `swImplPolicy` is set to `queued`, it is not required that all elements on the sender side need to be mapped to elements on the receiver side to achieve compatibility.] (*RS_SWCT_03210, RS_SWCT_03135*)

The details regarding the compatibility rules are explained in chapter 6.3.

[constr_1279] Unmapped elements of `ApplicationCompositeDataTypes` or `ImplementationDataTypes` and the attribute `swImplPolicy` [If the attribute `swImplPolicy` is set to `queued` it is not allowed to have unmapped elements of `ApplicationCompositeDataTypes` or `ImplementationDataTypes` of category `STRUCTURE` or `ARRAY` on the receiver side.] ()

[constr_1280] Unmapped `dataElement` on the receiver side shall have an `initValue` [If elements of `ApplicationCompositeDataTypes` or `ImplementationDataTypes` of category `STRUCTURE` or `ARRAY` are not considered in a `SubElementMapping` then the enclosing `dataElement` shall have an `initValue` if the `NonqueuedReceiverComSpec` is aggregated by an `AbstractRequiredPortPrototype`.] ()

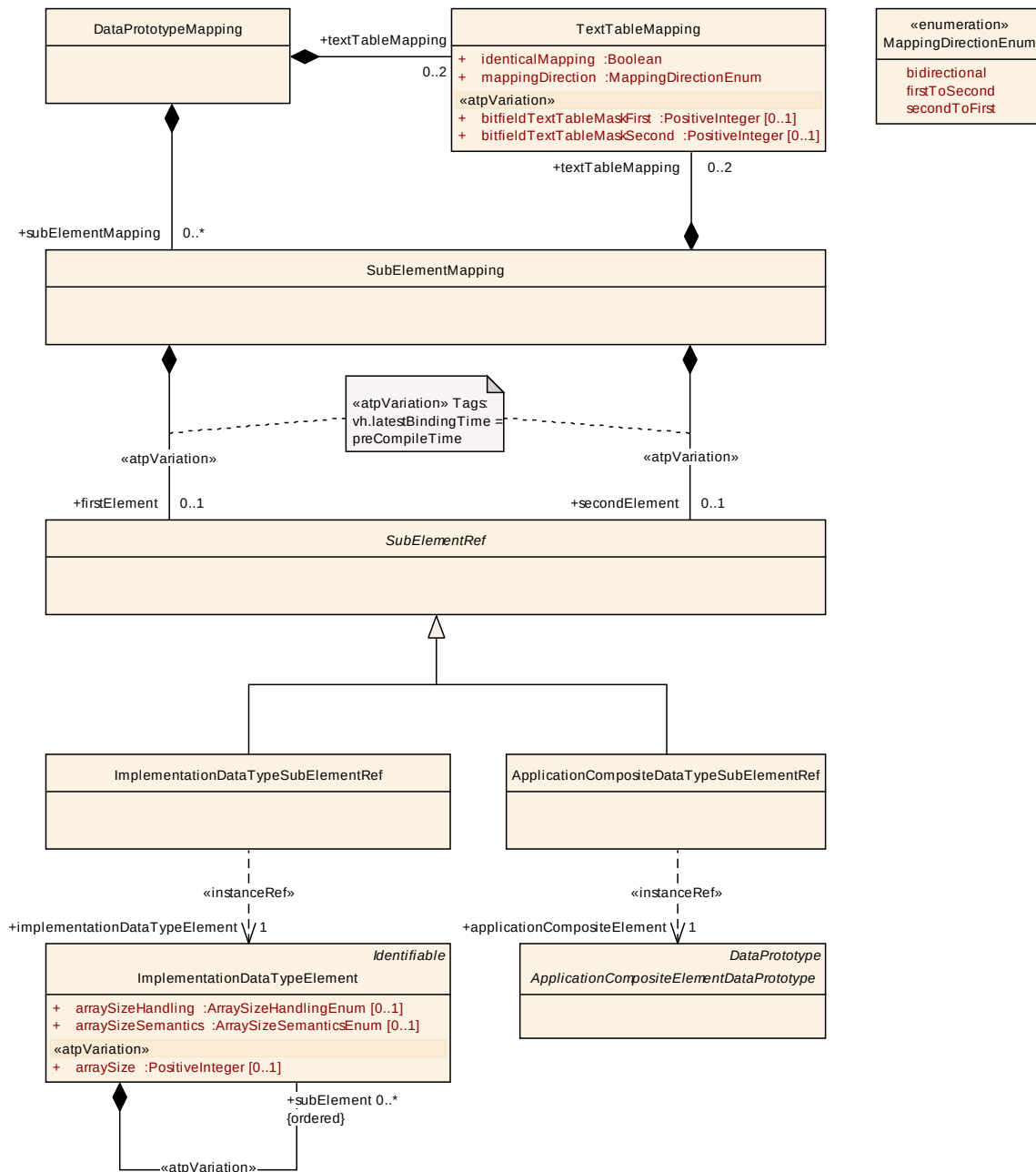


Figure 4.16: Mapping of elements of composite data types

[TPS_SWCT_01024] Combination of **ApplicationCompositeDataType** and nested **ImplementationDataType** [The mapping of elements of **ApplicationCompositeDataTypes** or **ImplementationDataTypes** of category **STRUCTURE** or **ARRAY** works for both **ApplicationCompositeDataType** and nested **ImplementationDataTypes** and even for combinations of them, i.e. one **PortInterface** may use an **ApplicationCompositeDataType** while the other **PortInterface** uses a nested **ImplementationDataType**.]([RS_SWCT_03210](#), [RS_SWCT_03135](#))

[TPS_SWCT_01195] Mapping of composite element to primitive **DataPrototype** [It is also possible to map an element of a composite data type on the provided side to

a primitive `DataPrototype` on the required side. For this purpose the multiplicity of the `firstElement` shall be set to 1 and the multiplicity of the `secondElement` shall be set to 0. `](RS_SWCT_03136)`

In general, the multiplicity of the `firstElement` can technically also be set to 0 but this case is reserved for future use.

[constr_1190] Only one mapping for composite to primitive use case `[` In the case described by `[TPS_SWCT_01195]` only one `subElementMapping` shall exist at the enclosing `DataPrototypeMapping`. `](`

[constr_1300] Primitive `DataPrototype` on the provider side shall not be mapped to element of a composite data type on the requester side `[` The usage of `DataPrototypeMapping` resp. `SubElementMapping` does not support the following configuration:

- The `AutosarDataPrototype` referenced on the provider/client side is typed by an `ApplicationPrimitiveDataType` of category `VALUE` or `ImplementationDataType` of category `VALUE` or category `TYPE_REFERENCE` that eventually resolves to category `VALUE`.
- The `DataPrototypeMapping` aggregates a `subElementMapping` that refers to a `ImplementationDataTypeElement` or `ApplicationCompositeElementDataPrototype` on the requester/server side.

`](`

Class	SubElementMapping			
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
Note	This meta-class allows for the definition of mappings of elements of a composite data type.			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
firstElement	<code>SubElementRef</code>	0..1	aggr	This represents the first element referenced in the scope of the mapping. Stereotypes: <code>atpVariation</code> Tags: <code>vh.latestBindingTime=preCompileTime</code>
secondElement	<code>SubElementRef</code>	0..1	aggr	This represents the second element referenced in the scope of the mapping. Stereotypes: <code>atpVariation</code> Tags: <code>vh.latestBindingTime=preCompileTime</code>
textTableMapping	<code>TextTableMapping</code>	0..2	aggr	This allows for the text-table translation of individual elements of a composite data type.

Table 4.34: SubElementMapping

Class	SubElementRef (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
Note	This meta-class provides the ability to reference elements of composite data type.			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
–	–	–	–	–

Table 4.35: SubElementRef

Class	ImplementationDataTypeSubElementRef			
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
Note	This meta-class represents the specialization of SubElementMapping with respect to ImplementationDataTypes.			
Base	ARObject, SubElementRef			
Attribute	Datatype	Mul.	Kind	Note
implementationDataTypeElement	ArVariableInImplementationDataInstanceRef	1	aggr	This represents the referenced implementationDataTypeElement.

Table 4.36: ImplementationDataTypeSubElementRef

Class	ApplicationCompositeDataTypeSubElementRef			
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
Note	This meta-class represents the specialization of SubElementMapping with respect to ApplicationCompositeDataTypes.			
Base	ARObject, SubElementRef			
Attribute	Datatype	Mul.	Kind	Note
applicationCompositeElement	ApplicationCompositeElementDataPrototype	1	iref	This represents the referenced ApplicationCompositeDataPrototype.

Table 4.37: ApplicationCompositeDataTypeSubElementRef

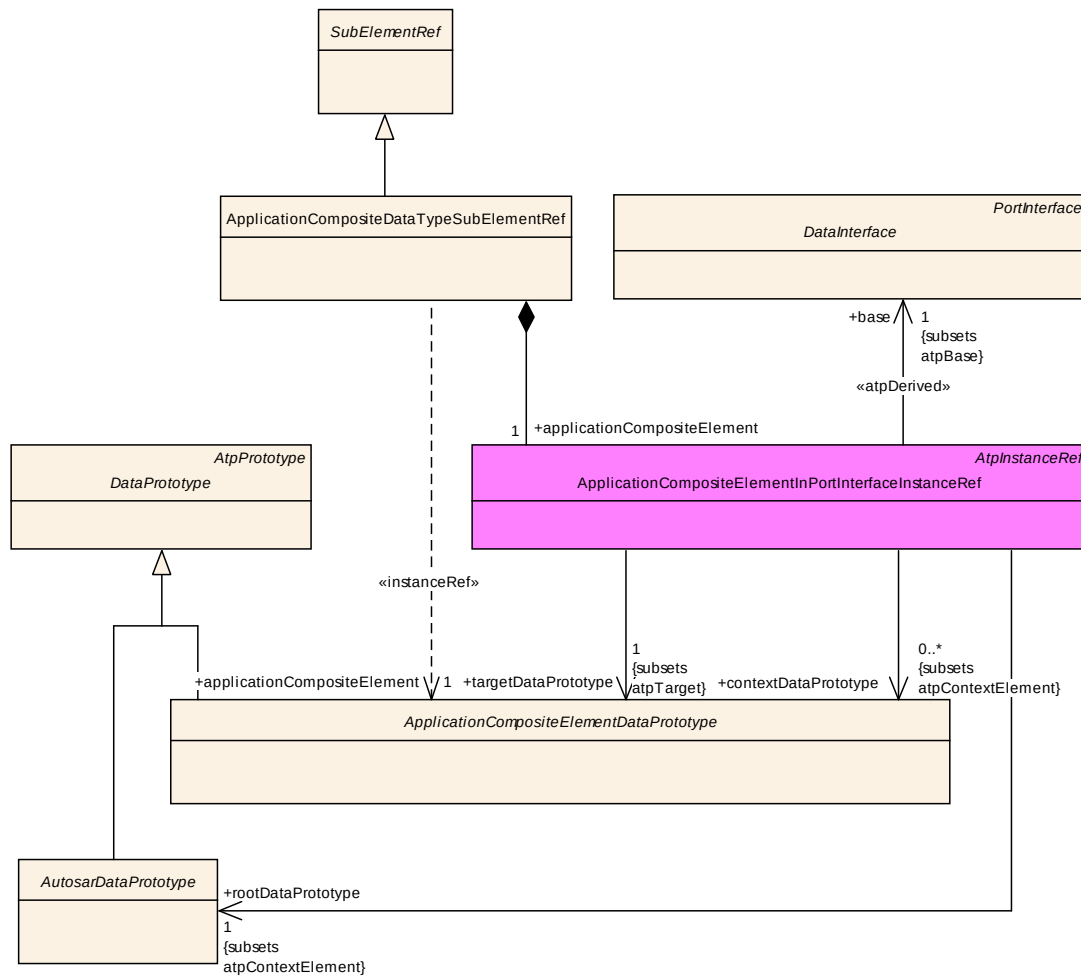


Figure 4.17: Implementation of the InstanceRef for the mapping of elements of composite application data types

[constr_1184] Consistency of `rootDataPrototype` and `base` in the context of `ApplicationCompositeElementInPortInterfaceInstanceRef` [The `rootDataPrototype` referenced by `ApplicationCompositeElementInPortInterfaceInstanceRef` shall be owned by the applicable subclass of `DataInterface` referenced in the role `base`. This implies that the `rootDataPrototype` shall be a `ParameterDataPrototype` if the `base` is a `ParameterInterface`. Otherwise the `rootDataPrototype` shall be a `VariableDataPrototype`.]()

[constr_1185] Consistency of data types in the context of `ApplicationCompositeElementInPortInterfaceInstanceRef` [The definition of attributes `contextDataPrototype` and `targetDataPrototype` shall (via the type-prototype pattern) be enclosed in the context of the definition of the data type used to type `rootDataPrototype`.]()

In other words, it shall be possible to reach `contextDataPrototype` and `targetDataPrototype` by means of the type-prototype chain created by the definition of the data type used to type `rootDataPrototype`. And, as implied by the definition of the InstanceRef, the `contextDataPrototypes` shall enclose each others and, eventually, the `targetDataPrototype`.

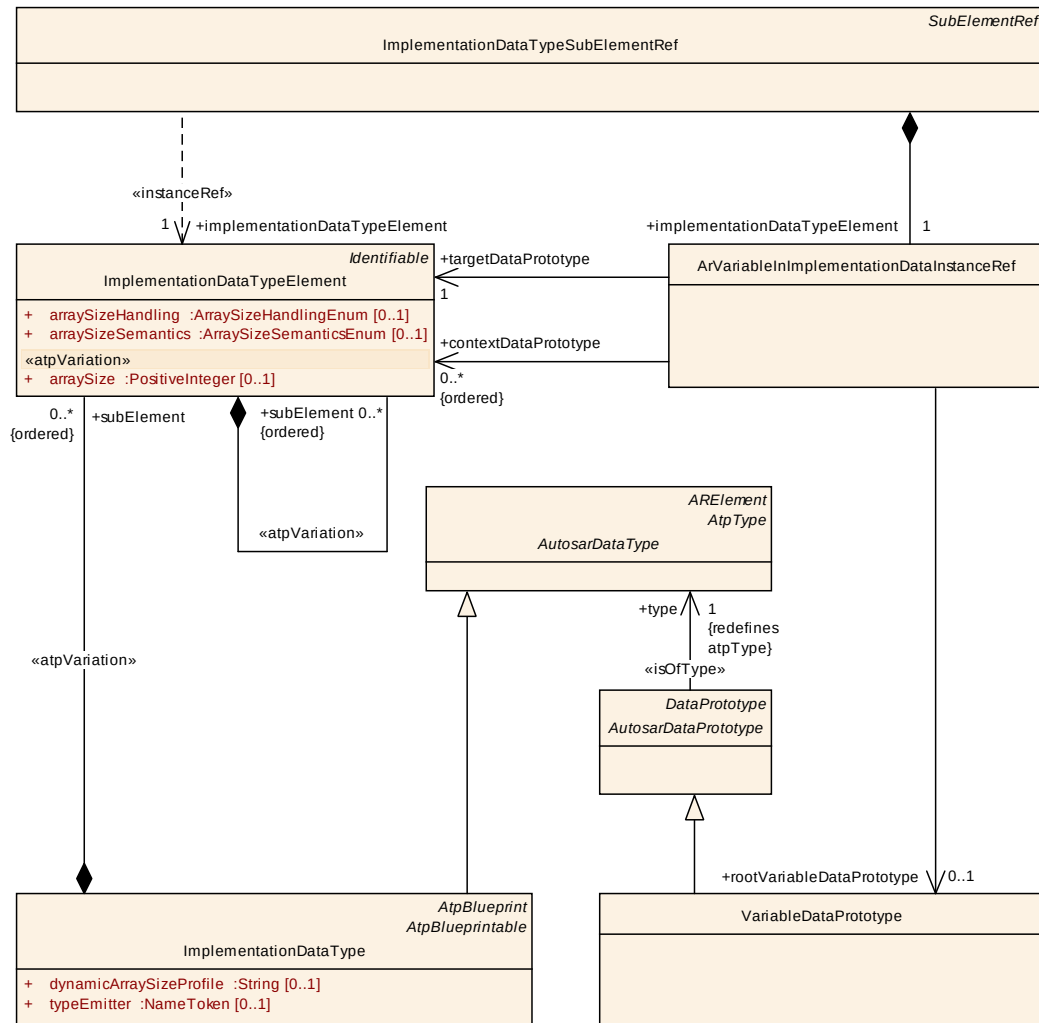


Figure 4.18: Implementation of the InstanceRef for the mapping of elements of composite implementation data types

[constr_1186] Consistency of data types in the context of **ArVariableInImplementationDataInstanceRef** [The definition of attributes `contextDataPrototype` and `targetDataPrototype` shall be enclosed in the context of the definition of the data type used to type `rootDataPrototype`.]()

4.3.2 Data Conversion

[TPS_SWCT_01560] Supported **categorys** of **CompuMethods** for data conversion [Data conversion shall be supported for **AutosarDataTypes** that refer to **CompuMethods** of **category** `LINEAR`, `IDENTICAL`, `SCALE_LINEAR_AND_TEXTTABLE`, `TEXTTABLE`, and `BITFIELD_TEXTTABLE`.](**RS_SWCT_03210**)

[TPS_SWCT_01561] Application of data conversion to composite **AutosarDataTypes** [Data conversion is also applicable for composite **AutosarDataTypes**. The actual conversion, however, shall be individually applied to each leaf element of a given composite **AutosarDataType**.](**RS_SWCT_03210**)

4.3.2.1 Linear Data Scaling

A *Linear Data Scaling* can be defined under following preconditions:

[TPS_SWCT_01549] Definition of linear data scaling [The term `Linear Scaling` is defined as follows:

1. Regarding the existence of `CompuMethods` one of the following cases shall apply:
 - (a) The involved `AutosarDataTypes` refer to `CompuMethods` of `category IDENTICAL, LINEAR, or RAT_FUNC`.
 - (b) If one side (sender or receiver) does not refer a `CompuMethod` then a “default” `CompuMethod` of `category IDENTICAL` shall be assumed.
2. Regarding the existence of `Units` one of the following cases shall apply:
 - (a) The `CompuMethods` refer either to compatible `Units` or to `Units` that in turn refer to compatible definitions of `PhysicalDimension`.
 - (b) `Units` and `PhysicalDimensions` do partially not exist on one side:
 - If one side (sender or receiver) does not refer to a `Unit`, then an “imaginary” `Unit` with the properties defined in [TPS_SWCT_01492] shall be assumed.
 - if the `PhysicalDimension` is only defined on one side (sender or receiver) then it shall be considered as default for the other side.
3. Both `CompuMethods` fulfill the following condition:

$$Int = \frac{N_0 * phys^0 + N_1 * phys^1 + N_2 * phys^2 + \dots + N_i * phys^i}{D_0 * phys^0 + D_1 * phys^1 + D_2 * phys^2 + \dots + D_i * phys^i} \text{ with}$$
 - $N_2 = N_3 = \dots = N_i = 0$
 - $D_1 = D_2 = \dots = D_i = 0$
 - $N_1 \neq 0$
 - $D_0 \neq 0$

The coefficient N_0 represents the offset and can take any value.

](RS_SWCT_03210)

[TPS_SWCT_01550] Definition of reciprocal linear data scaling [The term `Reciprocal Linear Scaling` is defined as follows:

1. The involved `AutosarDataTypes` refer to `CompuMethods` of `category RAT_FUNC`.
2. The `CompuMethods` refer either to compatible `Units` or to `Units` that in turn refer to compatible definitions of `PhysicalDimension`.

3. Both `CompuMethods` fulfill the following condition:

$$Int = \frac{N_0 * phys^0 + N_1 * phys^1 + N_2 * phys^2 + \dots + N_i * phys^i}{D_0 * phys^0 + D_1 * phys^1 + D_2 * phys^2 + \dots + D_i * phys^i} \text{ with}$$

- $N_1 = N_2 = \dots = N_i = 0$
- $D_2 = D_3 = \dots = D_i = 0$
- $N_0 \neq 0$
- $D_1 \neq 0$

The coefficient D_0 represents the (reciprocal) offset and can take any value.

]([RS_SWCT_03210](#))

[TPS_SWCT_01168] Linear conversion factor can be calculated [In such cases a linear conversion factor can be calculated out of the `factorSiToUnit` and `offsetSiToUnit` attributes of the referred `Units` and the `CompuRationalCoeffs` of a `compuInternalToPhys/compuPhysToInternal` of the referred `CompuMethods`.]([RS_SWCT_03210](#))

4.3.2.2 Table Conversion

[TPS_SWCT_01162] Existence of `TextTableMapping` [A `TextTableMapping` can be defined if the `AutosarDataTypes` refer to `CompuMethods` of category `TEXTTABLE`, `SCALE_LINEAR_AND_TEXTTABLE`, and `BITFIELD_TEXTTABLE`.]([RS_SWCT_03210](#))

Please note that the use case behind the appearance of `BITFIELD_TEXTTABLE` in **[TPS_SWCT_01162]** is the fact that BSW modules such as the `Dem` need to put data into the NVRAM that has the nature of single bits embedded into a composite data type.

The `TextTableMapping` is defined as a table based conversion.

[TPS_SWCT_01163] Conversion from `firstValue` to `secondValue` [A `firstValue` of a `valuePair` is converted into the `secondValue` in case of a data flow from the `firstDataPrototype` to the `secondDataPrototype`.]([RS_SWCT_03210](#))

[TPS_SWCT_01164] Conversion from `secondValue` to `firstValue` [In case of a data flow from the `secondDataPrototype` to `firstDataPrototype` the `secondValue` is substituted by the `firstValue`.]([RS_SWCT_03210](#))

[TPS_SWCT_01165] Invertible mapping [If the `mappingDirection` attribute is set to `bidirectional` then the `TextTableMapping` has to be invertible. This requires that the list of all `firstValues` and the list of all `secondValues` do not contain identical values inside a list.]([RS_SWCT_03210](#))

[TPS_SWCT_01166] Non-invertible mapping [For non-invertible `TextTableMapping`, a dedicated `TextTableMapping` for each direction can be defined.]([RS_SWCT_03210](#))

[constr_1303] Applicability of [TextTableMapping](#) depending on the value of [CompuMethod.category](#) [If a [DataPrototypeMapping](#) aggregates a [TextTableMapping](#) then only certain combinations of the value of the applicable [CompuMethod.category](#) are supported:

- [category](#) of [firstDataPrototype](#): [TEXTTABLE](#),
[category](#) of [secondDataPrototype](#): [TEXTTABLE](#)
- [category](#) of [firstDataPrototype](#): [SCALE_LINEAR_AND_TEXTTABLE](#),
[category](#) of [secondDataPrototype](#): [TEXTTABLE](#)
- [category](#) of [firstDataPrototype](#): [TEXTTABLE](#),
[category](#) of [secondDataPrototype](#): [SCALE_LINEAR_AND_TEXTTABLE](#)
- [category](#) of [firstDataPrototype](#): [BITFIELD_TEXTTABLE](#),
[category](#) of [secondDataPrototype](#): [TEXTTABLE](#)
- [category](#) of [firstDataPrototype](#): [TEXTTABLE](#),
[category](#) of [secondDataPrototype](#): [BITFIELD_TEXTTABLE](#)
- [category](#) of [firstDataPrototype](#): [BITFIELD_TEXTTABLE](#),
[category](#) of [secondDataPrototype](#): [BITFIELD_TEXTTABLE](#)

]()

To some extent, *bitfields* can be regarded as a hybrid between a primitive and a structured data type:

- On the one hand, a *bitfield* is defined in the context of a primitive [ImplementationDataType](#).
- On the other hand, by means of the definition of a [mask](#), it is possible to define **isolated parts** within the primitive [ImplementationDataType](#) that potentially can be totally independent from each other with respect to the semantics of the data that match the [mask](#).

In other words, the existence of semantically independent and potentially isolated parts within the primitive [ImplementationDataType](#) creates a **similar characteristic** as if the definitions of the isolated parts were created by means of defining primitive [ImplementationDataTypeElements](#) within the context of a composite [ImplementationDataType](#).

And because it is possible to regard the “mission statement” of a [DataPrototype](#) that refers to a [CompuMethod](#) of [category](#) [BITFIELD_TEXTTABLE](#) as to mimic the semantics of a structured data type it is also possible to apply some of the rules that are already in place for structured data types in this specific case as well.

This conclusion, in combination with the existence of [\[TPS_SWCT_01551\]](#), sets the stage for [\[TPS_SWCT_01583\]](#).

[TPS_SWCT_01583] Completeness of [TextTableMapping](#) is not a requirement
[If a [DataPrototypeMapping](#) contains one or more [TextTableMapping](#)(s) where

the `DataPrototype` on the **sender side** refers to a `CompuMethod` of category `BITFIELD_TEXTTABLE` it is **not** required that for each possible value and each possible bit mask on the sender side corresponding values on the receiver side are specified.
]([RS_SWCT_03210](#))

With respect to [[TPS_SWCT_01583](#)] it is still important to observe that within a single mask **all values on the sender side shall have a mapping** to the receiver side.

Otherwise the RTE generator would not be able to create mapping code that unambiguously takes care of mapping the correct values onto each other.

[constr_1313] Completeness of `TextTableMapping` for the values of a given bit mask on the sender side [If a `DataPrototypeMapping` contains one or more `TextTableMapping`(s) where the `DataPrototype` on the **sender side** refers to a `CompuMethod` of category `BITFIELD_TEXTTABLE` then all `DataPrototypeMapping.textTableMapping` shall aggregate a collection of `TextTableMapping.valuePair` where each possible value of the **sender bit mask**⁵ is represented by exactly one `TextTableValuePair.firstValue` ([TPS_SWCT_01163](#)) resp. `TextTableValuePair.secondValue` ([TPS_SWCT_01164](#)).]()

[constr_1304] Existence of attribute `bitfieldTextTableMaskFirst` [The attribute `bitfieldTextTableMaskFirst` shall be defined **only if** the `firstDataPrototype` of a `DataPrototypeMapping` refers to a `CompuMethod` that has the value of `category` set to `BITFIELD_TEXTTABLE`.]()

[constr_1305] Existence of attribute `bitfieldTextTableMaskSecond` [The attribute `bitfieldTextTableMaskSecond` shall be defined **only if** the `secondDataPrototype` of a `DataPrototypeMapping` refers to a `CompuMethod` that has the value of `category` set to `BITFIELD_TEXTTABLE`.]()

[constr_1306] Limitation of `TextTableMapping` for `CompuMethods` that have the value of `category` set to `BITFIELD_TEXTTABLE` [For any `TextTableMapping` where both `firstDataPrototype` and `secondDataPrototype` refer to `CompuMethods` that have the value of `category` set to `BITFIELD_TEXTTABLE` and where the attribute `TextTableMapping.valuePair` exists the value of attribute `TextTableMapping.identicalMapping` shall be set to `false`.]()

[constr_1307] Consistency of values and masks in `TextTableMapping` [If a `TextTableMapping` element defines bit masks as `bitfieldTextTableMaskFirst` or `bitfieldTextTableMaskSecond` then all contained `TextTableMapping.valuePair.firstValues` as well as all `TextTableMapping.valuePair.secondValues` shall **not** specify a value that would be ruled out when - depending on the given value of `TextTableMapping.mappingDirection` - the relevant bit mask is applied.]()

Example for [[constr_1307](#)]: For a bit mask `0b00001000` only the corresponding values 8 and 0 are allowed.

⁵Depending on the applicable case this means either `bitfieldTextTableMaskFirst` (applies if [[TPS_SWCT_01163](#)] is in place) or `bitfieldTextTableMaskSecond` for the case of [[TPS_SWCT_01164](#)].

Class	TextTableMapping			
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
Note	Defines the mapping of two DataPrototypes typed by AutosarDataTypes that refer to CompuMethods of category TEXTTABLE, SCALE_LINEAR_AND_TEXTTABLE or BITFIELD_TEXTTABLE.			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
bitfieldText TableMask First	PositiveInteger	0..1	attr	This attribute can be used to support the mapping of bit field to bit field, boolean values to bit fields, and vice versa. The attribute defines the bit mask for the first element of the TextTableMapping. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime
bitfieldText TableMask Second	PositiveInteger	0..1	attr	This attribute can be used to support the mapping of bit field to bit field, boolean values to bit fields, and vice versa. The attribute defines the bit mask for the second element of the TextTableMapping. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime
identicalM apping	Boolean	1	attr	If identicalMapping is set == true the values of the two referenced DataPrototypes do not need any conversion of the values.
mappingDi rection	MappingDirectionEnum	1	attr	Specifies the conversion direction for which the TextTableMapping is applicable.
valuePair	TextTableValuePair	*	aggr	Defines a pair of values which are translated into each other.

Table 4.38: TextTableMapping

Enumeration	MappingDirectionEnum
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface
Note	Specifies the conversion direction for which the mapping is applicable.
Literal	Description
bidirectional	The TextTableMapping is applicable in both directions.
firstToSecond	The TextTableMapping is applicable in the direction from firstDataPrototype / firstOperationArgument referring into the PortInterface of the PPortPrototype to secondDataPrototype / secondOperationArgument referring into the PortInterface of the RPortPrototype.
secondTo First	The TextTableMapping is applicable in the direction from secondDataPrototype / secondOperationArgument referring into the PortInterface of the PPortPrototype to firstDataPrototype / firstOperationArgument referring into the PortInterface of the RPortPrototype.

Table 4.39: MappingDirectionEnum

Class	TextTableValuePair			
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
Note	Defines a pair of text values which are translated into each other.			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
firstValue	Numerical	1	attr	Value of first DataPrototype provided similar to a numerical ValueSpecification which is intended to be assigned to a Primitive data element. Note that the numerical value is a variant, it can be computed by a formula. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime
secondValue	Numerical	1	attr	Value of second DataPrototype provided similar to a numerical ValueSpecification which is intended to be assigned to a Primitive data element. Note that the numerical value is a variant, it can be computed by a formula. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime

Table 4.40: TextTableValuePair

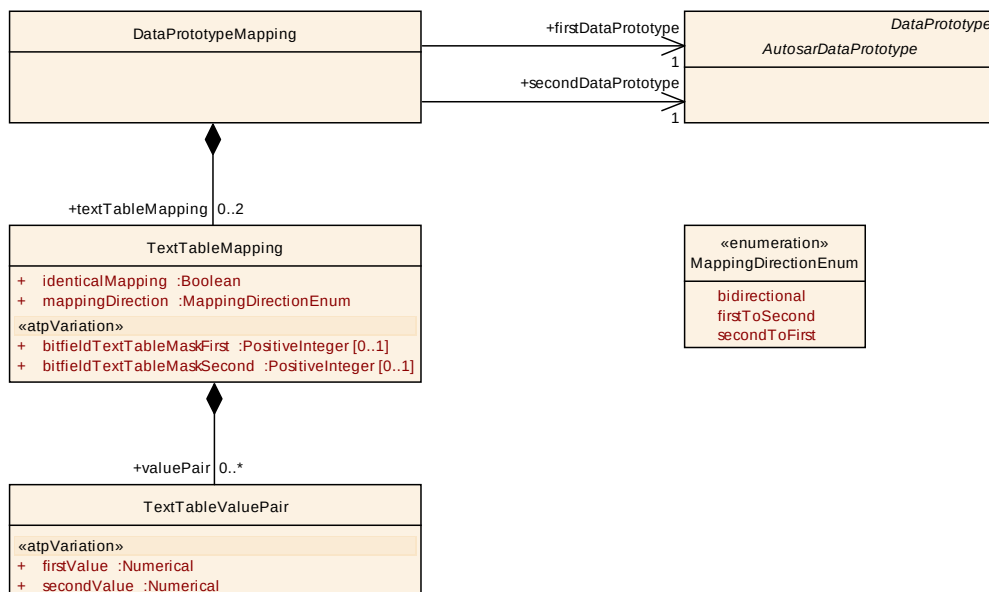


Figure 4.19: Mapping of DataPrototypes that eventually refer to CompuMethods of category TEXTTABLE, SCALE_LINEAR_AND_TEXTTABLE, and BITFIELD_TEXTTABLE

4.4 Port Annotation

4.4.1 Introduction

[TPS_SWCT_01203] [PortPrototype](#) may own port annotations [In addition to the formal specification required to implement the communication via ports, a [PortPrototype](#) may own so-called port annotations (please find a summary in Figure 4.20). They do not directly influence the signature of calls via this [PortPrototype](#), but contain further information that may be useful for the application developers of the components on both sides of the connection.]()

[TPS_SWCT_01204] [GeneralAnnotation](#) [Beside formally specified attributes it is also possible to place textual information as provided in [GeneralAnnotation](#).]()

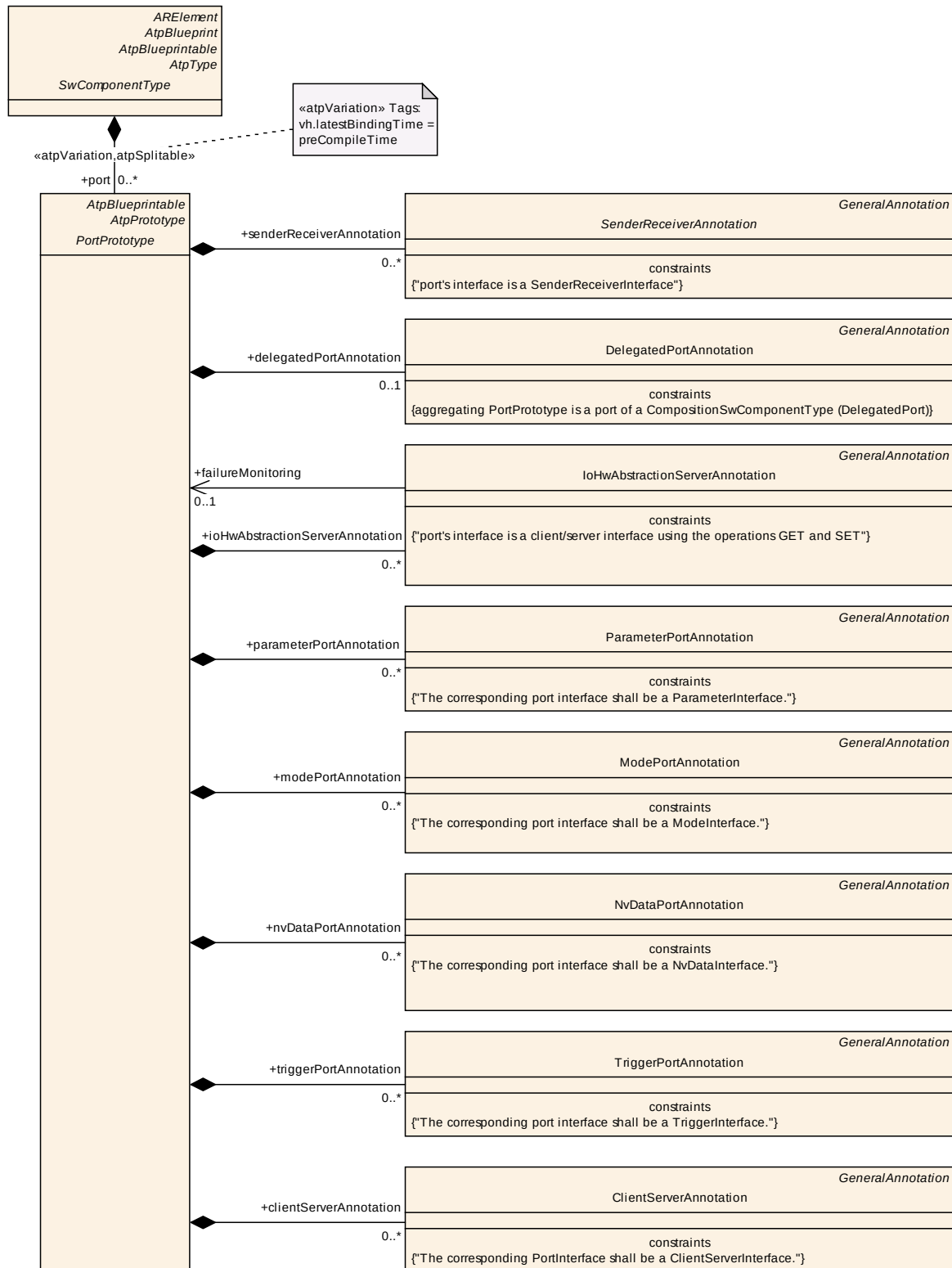


Figure 4.20: Application Level Port Annotations Overview

4.4.2 SenderReceiverAnnotation

Embedded automotive software is used to implement open-loop and closed-loop control-algorithms. Therefore, a software-component description has to accommodate typical control engineering description means which have only indirect influence of the embedded software itself.

These annotations provide the (function-) developer with a direct indication whether a certain software-component is appropriate for the control-algorithm to be designed. A typical annotation is the signal quality which is characterized by several properties. Each of the property is an annotation in its own.

[TPS_SWCT_01205] Typical annotations for sender/receiver communication [Typical annotations for sender/receiver communication are:

- **Signal Age:** this attribute expresses that the associated software-component will only work correctly given that the propagation of the signal from a sensor to a consumer can be finished within a particular time-limit. Of course, this cannot be identified on component or role level, but has to take into account the instance view as well as the actual ECU- and bus-scheduling.
- **Raw:** a raw signal is typically taken directly from the basic software modules of the ECU abstraction layer. In particular, no sensor software-component has filtered its original value. A `dataElement` in an `RPortPrototype` of a `SwComponentType` using this annotation indicates to the control engineer (who develops a control-algorithm for this component) that the signal has to be filtered (This relationship applies for `SenderReceiverInterfaces`).
- **Filtered:** this attribute indicates that a raw signal has been manipulated by some application software-components by using a certain filter.
- **Computed:** this attribute indicates that this signal is not measured directly but calculated from tentatively several other measured or calculated signals. In a vehicle, there might be alternative signals to be used from other components having a better quality, e.g. a raw signal.
- **Min:** this annotation indicates that the signal carries a minimum value. If, for example, a reference value computed in the software-component is below that value some dedicated actions (e.g. failure-mode) might have to be taken.
- **Max:** this annotation indicates that the signal carries a maximum value. If, for example, a reference value computed in the software-component is above that value some dedicated actions (e.g. failure-mode) might have to be taken.

In the meta-model this aspect is implemented by the abstract meta-class `SenderReceiverAnnotation` which represents the base class of both `SenderAnnotation` and `ReceiverAnnotation`. This relationship is depicted in Figure 4.21. `]()`

Class	SenderReceiverAnnotation (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::ApplicationAttributes			
Note	Annotation of the data elements in a port that realizes a sender/receiver interface.			
Base	ARObject, GeneralAnnotation			
Attribute	Datatype	Mul.	Kind	Note
computed	Boolean	1	attr	Flag whether this data element was not measured directly but instead was calculated from possibly several other measured or calculated values.
dataElement	VariableDataPrototype	1	ref	The instance of VariableDataPrototype annotated.
limitKind	DataLimitKindEnum	1	attr	This min or max has not to be mismatched with the min- and max for data-value in a compu-method. For example, this annotation shows when the result of the calculation performed in a RunnableEntity owned by one AtomicSwComponentType is transmitted to another AtomicSwComponentType whose RunnableEntity will use this value as a limit, e.g. the max.power which can be used by that software-component, or the current min. slip.
processingKind	ProcessingKindEnum	1	attr	This attribute controls how data is processed according to the possible values of ProcessingKindEnum.

Table 4.41: SenderReceiverAnnotation

Class	SenderAnnotation			
Package	M2::AUTOSARTemplates::SWComponentTemplate::ApplicationAttributes			
Note	Annotation of a sender port, specifying properties of data elements that don't affect communication or generation of the RTE.			
Base	ARObject, GeneralAnnotation , SenderReceiverAnnotation			
Attribute	Datatype	Mul.	Kind	Note
—	—	—	—	—

Table 4.42: SenderAnnotation

Class	ReceiverAnnotation			
Package	M2::AUTOSARTemplates::SWComponentTemplate::ApplicationAttributes			
Note	Annotation of a receiver port, specifying properties of data elements that don't affect communication or generation of the RTE. The given attributes are requirements on the required data.			
Base	ARObject, GeneralAnnotation , SenderReceiverAnnotation			
Attribute	Datatype	Mul.	Kind	Note
signalAge	MultidimensionalTime	1	aggr	The maximum allowed age of the signal since it was originally read by a sensor. This is a requirement specified on the receiver side.

Table 4.43: ReceiverAnnotation

<i>Enumeration</i>	ProcessingKindEnum
Package	M2::AUTOSARTemplates::SWComponentTemplate::ApplicationAttributes
Note	Kind of processing which has been applied to a data element.
Literal	Description
filtered	Indicates that a raw signal has been manipulated by some application software components by using filters.
none	Indicates that none of the other option apply.
raw	Specifies that a signal is taken directly from the basic software modules, i.e. from the ECU abstraction layer. It indicates to a developer that the control algorithm in the software has to provide filters.

Table 4.44: ProcessingKindEnum

<i>Enumeration</i>	DataLimitKindEnum
Package	M2::AUTOSARTemplates::SWComponentTemplate::ApplicationAttributes
Note	Indicates whether the data element carries a minimum or maximum value, thereby limiting the current range of another value.
Literal	Description
max	Limitation to maximum value
min	Limitation to minimum value
none	No limitation applicable

Table 4.45: DataLimitKindEnum

[TPS_SWCT_01206] Min and Max annotations are valid for a certain amount of time [The Min and Max annotations are valid for a certain amount of time. The value is likely to change to another valid value while the ECU is running. E.g. the maximal torque which can be requested from an engine is a typical use-case.]()

This value might vary depending on e.g. the status of the climate control system. Therefore, these annotations shall not be mismatched with the min and max attributes of [CompuMethods](#).

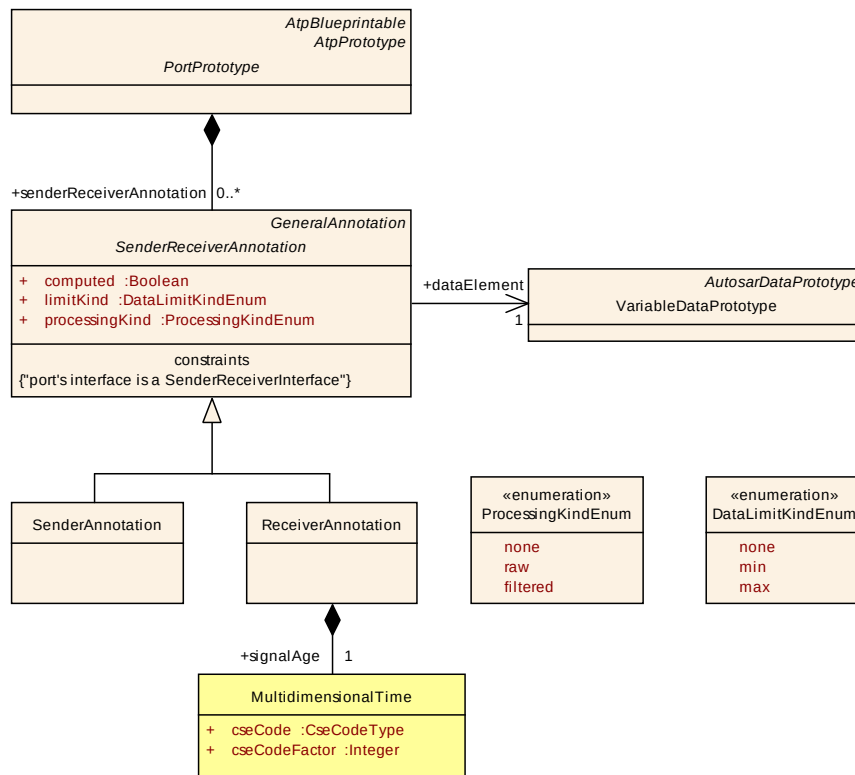


Figure 4.21: **SenderReceiverAnnotation**

The application level port annotations for sender/receiver communication have to be associated to each **dataElement** in a **PortPrototype**, e.g. there might be a “raw” **dataElement** and a “filtered” **dataElement** in the same **PortPrototype**!

[TPS_SWCT_01207] VariableDataPrototypes use the same application-level SenderReceiverAnnotation [Furthermore, if two **VariableDataPrototypes** use the same application-level **SenderReceiverAnnotation**, a reference from the annotation to the **VariableDataPrototypes** will be established by an appropriate tool.]()

[TPS_SWCT_01208] Grouping for SenderReceiverAnnotation [As shown in Figure 4.21 the **SenderReceiverAnnotation** for sender/receiver communication are grouped into

- processing type, indicating to some extend the direct quality of the signal,
- computed, which is just a flag or,
- limit type, showing the component expects an actual limit.

In the case of an **RPortPrototype**, the signal age of the value, carried by the associated **SwConnector**, can be specified. Each of these groups can be interpreted as a property of the signal-quality.]()

[constr_4004] Context of SenderReceiverAnnotation [A **SenderReceiverAnnotation** shall only be aggregated by a **PortPrototype** typed by a **SenderReceiverInterface**.]()

4.4.3 ClientServerAnnotation

[TPS_SWCT_01209] **ClientServerAnnotation** [The **ClientServerAnnotation** can be used to provide more information with respect to the **ClientServerOperation** of the **PortPrototype**.]()

Class	ClientServerAnnotation			
Package	M2::AUTOSARTemplates::SWComponentTemplate::ApplicationAttributes			
Note	Annotation to a port regarding a certain Operation.			
Base	ARObject, GeneralAnnotation			
Attribute	Datatype	Mul.	Kind	Note
operation	ClientServerOperation	1	ref	This represents the ClientServerOperation that the ClientServerAnnotation corresponds to.

Table 4.46: ClientServerAnnotation

The main use-case is to allow define additional information related to the **ClientServerOperation**.

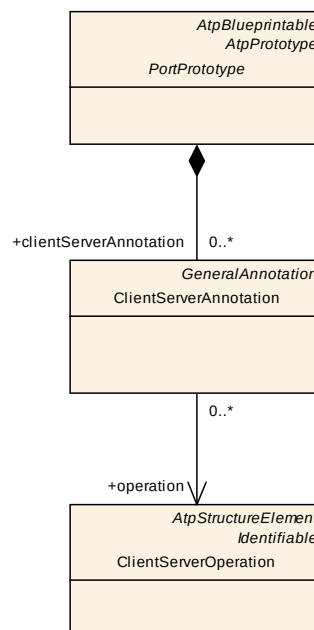


Figure 4.22: ClientServerAnnotation

[constr_4005] **Context of ClientServerAnnotation** [A **ClientServerAnnotation** shall only be aggregated by a **PortPrototype** typed by a **ClientServerInterface**.]()

4.4.4 Annotation for the I/O Hardware Abstraction Layer

Within the ECU-Abstraction Layer there are ECU-signals defined. These signals represent the electrical signals as they arrive in the micro-controller peripheral and are fetched from the registers via the MCAL.

Access to the I/O Hardware Abstraction Layer is done via service interfaces, i.e. the I/O Hardware Abstraction Layer provides GET- and SET-operations at the specified service ports of a [SensorActuatorSwComponentType](#).

[TPS_SWCT_01524] Usage of [IoHwAbstractionServerAnnotation](#) [[IoHwAbstractionServerAnnotation](#) can be used for all kinds of [PortInterfaces](#) except [NvDataInterface](#).]()

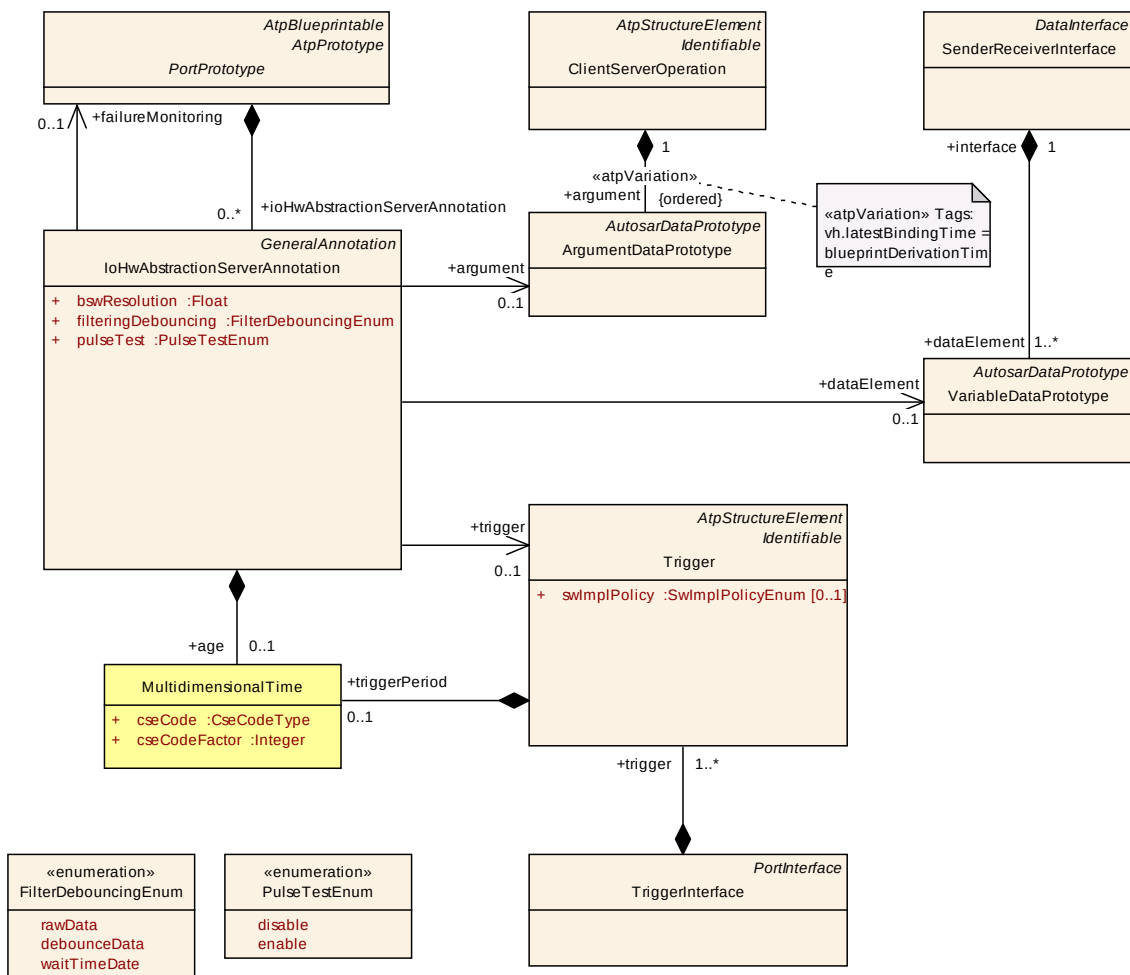


Figure 4.23: [IoHwAbstractionServerAnnotation](#)

Class	IoHwAbstractionServerAnnotation			
Package	M2::AUTOSARTemplates::SWComponentTemplate::ApplicationAttributes			
Note	The IoHwAbstractionServerAnnotation will only be used from a sensor- or an actuator component while interacting with the IoHwAbstraction layer. Note that the "server" in the name of this meta-class is not meant to restrict the usage to ClientServerInterfaces.			
Base	ARObject, GeneralAnnotation			
Attribute	Datatype	Mul.	Kind	Note
age	MultidimensionalTime	0..1	aggr	In case of a SET operation, the age will be interpreted as Delay while in a GET operation (input) it specifies the Lifetime of the signal within the IoHwAbstraction Layer Tags: xml.sequenceOffset=10
argument	ArgumentDataPrototype	0..1	ref	Reference to the corresponding ArgumentDataPrototype. Tags: xml.sequenceOffset=20
bswResolution	Float	1	attr	This value is determined by an appropriate combination of the range, the unit as well as the data-elements type, i.e. $(\text{ecuSignalRange.upperLimit} - \text{ecuSignalRange.lowerLimit}) / (2^{\text{datatypeLength}} - 1)$ Tags: xml.sequenceOffset=30
dataElement	VariableDataPrototype	0..1	ref	Reference to the corresponding VariableDataPrototype. Tags: xml.sequenceOffset=40
failureMonitoring	PortPrototype	0..1	ref	This is only applicable in SET operations. If it is enabled, the IoHwAbstraction layer will monitor the result of the operation and issue an diagnostic signal. This means especially, that an additional client-server port has to be created. Tools can use this information to cross-check whether for each data-element in a SET operation with FailureMonitoring enabled an additional port is created The referenced port monitors a failure in the to be monitored VariableDataPrototype of the IoHwAbstraction layer. The referenced port has to be another port of the same Actuator or Sensor Component. Tags: xml.sequenceOffset=50

Attribute	Datatype	Mul.	Kind	Note
filteringDe bouncing	FilterDebouncin gEnum	1	attr	This attribute is used to indicate what kind of filtering/debouncing has been put to the signal in the IoHwAbstraction layer. rawData means that no modification of the signal has been applied. This is the default value debounceData means that the signal is a mean value waitTimeData means that the signal is delivered by a GET operation after a certain amount of time Tags: xml.sequenceOffset=60
pulseTest	PulseTestEnum	1	attr	This attribute indicates to the connected SensorActuatorSwComponentType whether the VariableDataPrototype can be used to generate pulse test sequences using the IoHwAbstraction layer Tags: xml.sequenceOffset=70
trigger	Trigger	0..1	ref	Reference to the corresponding Trigger. Tags: xml.sequenceOffset=80

Table 4.47: IoHwAbstractionServerAnnotation

Enumeration	FilterDebouncingEnum
Package	M2::AUTOSARTemplates::SWComponentTemplate::ApplicationAttributes
Note	This enumeration defines possible values for the filter debouncing strategy.
Literal	Description
debounce Data	The signal is a mean value
rawData	Means that no modification of the signal has been applied. This is the default value
waitTimeDate	The signal is delivered by a GET operation after a certain amount of time

Table 4.48: FilterDebouncingEnum

Enumeration	PulseTestEnum
Package	M2::AUTOSARTemplates::SWComponentTemplate::ApplicationAttributes
Note	This element indicates to the connected Actuator Software component whether the data-element can be used to generate pulse test sequences using the IoHwAbstraction layer
Literal	Description
disable	Disables the pulse test
enable	Enables the pulse test

Table 4.49: PulseTestEnum

[TPS_SWCT_01211] Assign several annotations to [ArgumentDataPrototype](#) [The [ClientServerOperations](#) provide an [ArgumentDataPrototype](#) where sev-

eral annotations can be assigned to. They are depicted in the [IoHwAbstraction-ServerAnnotation](#) meta-class in Figure 4.23. `()`

A detailed description of the attributes can be found in the IoHwAbstraction Layer software specification document [17]. For example, the signal age has a very dedicated meaning in this particular interface with respect to a register whereas the signal age in the [SenderReceiverAnnotation](#) is more generic. Especially, there is no relationship with the micro-controller peripherals.

4.4.5 Parameter Port Annotation

[TPS_SWCT_01212] [ParameterPortAnnotation](#) [The [ParameterPortAnnotation](#) can be used to provide more information with respect to calibration parameter prototypes of the [PortPrototype](#). The data provided at the [PortPrototype](#) is calibration parameters. The [ParameterPortAnnotation](#) provides a reference to a particular [ParameterDataPrototype](#). `()`

Class	ParameterPortAnnotation			
Package	M2::AUTOSARTemplates::SWComponentTemplate::ApplicationAttributes			
Note	Annotation to a port used for calibration regarding a certain ParameterDataPrototype.			
Base	ARObject, GeneralAnnotation			
Attribute	Datatype	Mul.	Kind	Note
parameter	ParameterDataPrototype	1	ref	The instance of annotated ParameterDataPrototype.

Table 4.50: ParameterPortAnnotation

The main use-case is to allow easy access to the information which calibration parameters influence the data on the [PortPrototype](#).

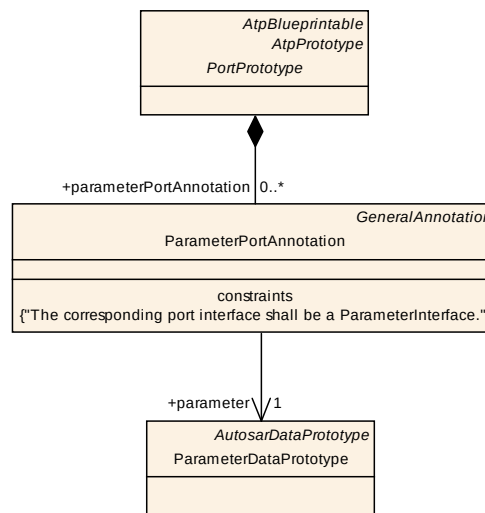


Figure 4.24: [ParameterPortAnnotation](#)

[constr_4006] Context of [ParameterPortAnnotation](#) [A [ParameterPortAnnotation](#) shall only be aggregated by a [PPortPrototype](#) owned by a [ParameterSwComponentType](#).]()

4.4.6 Mode Port Annotation

[TPS_SWCT_01213] [ModePortAnnotation](#) [The [ModePortAnnotation](#) can be used to provide more information with respect to the mode declaration group prototype of the [PortPrototype](#).]()

Class	ModePortAnnotation			
Package	M2::AUTOSARTemplates::SWComponentTemplate::ApplicationAttributes			
Note	Annotation to a port used for calibration regarding a certain ModeDeclarationGroupPrototype.			
Base	ARObject, GeneralAnnotation			
Attribute	Datatype	Mul.	Kind	Note
modeGroup	ModeDeclarationGroupPrototype	1	ref	The instance of annotated ModeDeclarationGroupPrototype.

Table 4.51: ModePortAnnotation

The main use-case is to allow for the definition of additional information related to the mode declaration group prototype.

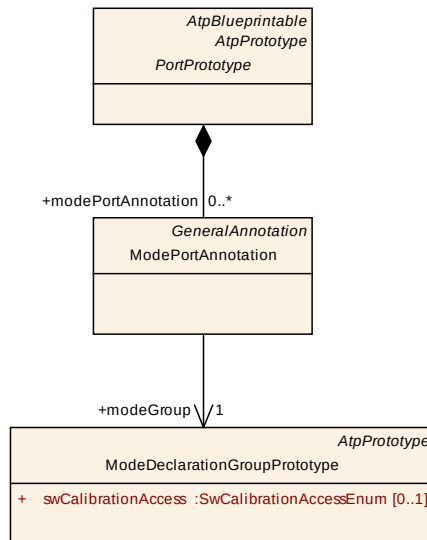


Figure 4.25: ModePortAnnotation

[constr_4007] Context of [ModePortAnnotation](#) [A [ModePortAnnotation](#) shall only be aggregated by a [PortPrototype](#) typed by a [ModeSwitchInterface](#).]()

4.4.7 Trigger Port Annotation

[TPS_SWCT_01214] **TriggerPortAnnotation** [The `TriggerPortAnnotation` can be used to provide more information with respect to the trigger of the `PortPrototype`.]()

Class	TriggerPortAnnotation			
Package	M2::AUTOSARTemplates::SWComponentTemplate::ApplicationAttributes			
Note	Annotation to a port used for calibration regarding a certain Trigger.			
Base	ARObject, GeneralAnnotation			
Attribute	Datatype	Mul.	Kind	Note
trigger	Trigger	1	ref	The instance of annotated trigger.

Table 4.52: TriggerPortAnnotation

The main use-case is to allow define additional information related to the trigger.

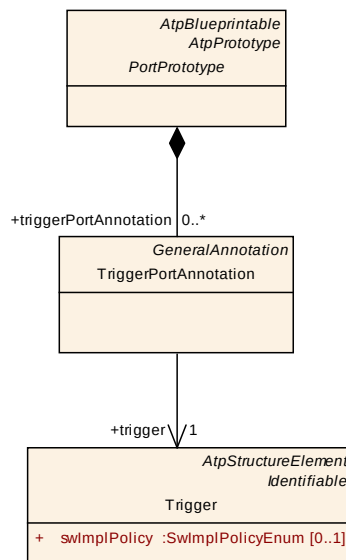


Figure 4.26: TriggerPortAnnotation

[constr_4008] **Context of TriggerPortAnnotation** [A `TriggerPortAnnotation` shall only be aggregated by a `PortPrototype` typed by a `TriggerInterface`.]()

4.4.8 Non Volatile Data Port Annotation

[TPS_SWCT_01215] **NvDataPortAnnotation** [The `NvDataPortAnnotation` can be used to provide more information with respect to the non volatile data of the `PortPrototype`.]()

Class	NvDataPortAnnotation			
Package	M2::AUTOSARTemplates::SWComponentTemplate::ApplicationAttributes			
Note	Annotation to a port regarding a certain VariableDataPrototype.			
Base	ARObject, GeneralAnnotation			
Attribute	Datatype	Mul.	Kind	Note
variable	VariableDataPrototype	1	ref	The instance of nv data annotated.

Table 4.53: NvDataPortAnnotation

The main use-case is to allow define additional information related to the non volatile data elements.

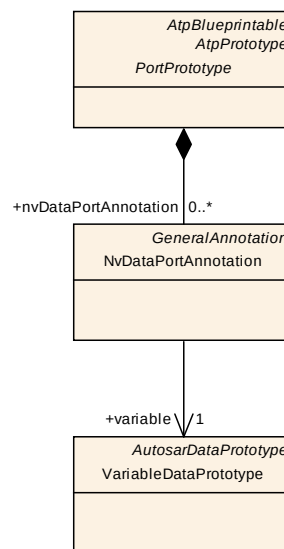


Figure 4.27: NvDataPortAnnotation

[constr_4009] Context of [NvDataPortAnnotation](#) [An [NvDataPortAnnotation](#) shall only be aggregated by a [PortPrototype](#) typed by an [NvDataInterface](#).]()

4.4.9 Delegated Port Annotations

[TPS_SWCT_01216] [DelegatedPortAnnotation](#) [The [DelegatedPortAnnotation](#) is used to define the Signal Fan In or Signal Fan Out inside the [CompositionSwComponentType](#).

This information is used to pre-define and pre-check resulting communication patterns in the VFB (1:n, n:1, 1:1) if empty [CompositionSwComponentTypes](#) are used as interface definition for sub-systems.

The [DelegatedPortAnnotation](#) guides either the system designer in connecting the empty [CompositionSwComponentType](#) or the sub-system designer in applying

communication pattern (1:n, n:1, 1:1) inside of the [CompositionSwComponentType](#).
]()

Class	DelegatedPortAnnotation			
Package	M2::AUTOSARTemplates::SWComponentTemplate::ApplicationAttributes			
Note	Annotation to a "delegated port" to specify the Signal Fan In or Signal Fan Out inside the CompositionSwComponentType.			
Base	ARObject, GeneralAnnotation			
Attribute	Datatype	Mul.	Kind	Note
signalFan	SignalFanEnum	0..1	attr	Specifies the Signal Fan In or Signal Fan Out inside the Composition Type.

Table 4.54: DelegatedPortAnnotation

Enumeration	SignalFanEnum
Package	M2::AUTOSARTemplates::SWComponentTemplate::ApplicationAttributes
Note	Signal Fan inside the Composition Component Type.
Literal	Description
ifold	The connections internally in the CompositionSwComponentType via DelegationSwConnectors and AssemblySwConnectors are defined in a way that at least one data element present in the S/R interface or one ClientServerOperation in the C/S interface of the outer PortPrototype is involved in a 1:n or n:1 communication pattern.
single	The connections internally in the CompositionSwComponentType via DelegationSwConnectors and AssemblySwConnectors are defined in a way that each VariableDataPrototype present in the S/R interface or ClientServerOperation in the C/S interface of the outer PortPrototype is involved in a 1:1 communication pattern only.

Table 4.55: SignalFanEnum

[TPS_SWCT_01217] **Semantics of [DelegatedPortAnnotation.signalFan](#)** [The attribute values have following definition:

- **single**: the internal connections in the [CompositionSwComponentType](#) via [DelegationSwConnectors](#) and [AssemblySwConnectors](#) are defined in a way that each [dataElement](#) present in the [SenderReceiverInterfaces](#) or [operation](#) in the [ClientServerInterfaces](#) of the outer [PortPrototype](#) is involved in a 1:1 communication pattern only.
- **ifold**: The internal connections in the [CompositionSwComponentType](#) via [DelegationSwConnectors](#) and [AssemblySwConnectors](#) are defined in a way that at least one [dataElement](#) present in the [SenderReceiverInterfaces](#) or one [operation](#) in the [ClientServerInterfaces](#) of the outer [PortPrototype](#) is involved in a 1:n or n:1 communication pattern.

]()

[constr_4010] Context of **DelegatedPortAnnotation** [A **DelegatedPortAnnotation** shall only be aggregated by a **PortPrototype** aggregated by a **CompositionSwComponentType**.]()

4.4.10 General Annotation

Besides formally specified attributes it is also possible to place textual information as provided in the abstract **GeneralAnnotation** (see Figure 4.28 for an overview).

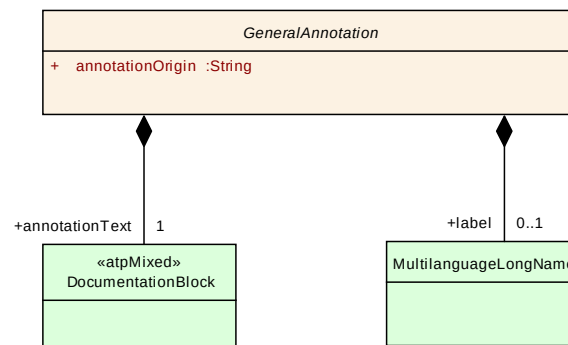


Figure 4.28: textual information in annotations

Class	GeneralAnnotation (abstract)			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::General Annotation			
Note	This class represents textual comments (called annotations) which relate to the object in which it is aggregated. These annotations are intended for use during the development process for transferring information from one step of the development process to the next one. The approach is similar to the "yellow pads" ... This abstract class can be specialized in order to add some further formal properties.			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
annotation Origin	String	1	attr	This attribute identifies the origin of the annotation. It is an arbitrary string since it can be an individual's name as well as the name of a tool or even the name of a process step. Tags: xml.sequenceOffset=30
annotation Text	Documentation Block	1	aggr	This is the text of the annotation. Tags: xml.sequenceOffset=40
label	MultilanguageLongName	0..1	aggr	This is the headline for the annotation. Tags: xml.sequenceOffset=20

Table 4.56: GeneralAnnotation

4.5 Communication Specification

[TPS_SWCT_01218] Big picture of ComSpec [The highest level of description of information exchanged between components in an AUTOSAR system is the [PortInterfaces](#), as shown in earlier sections. Such [PortInterface](#) however, only describes structure and does not include information about whether communication needs to be done reliably, or whether an initial value exists in case the real data is not yet available.

This information is role-specific, i.e. it shall be applied on the level of [PortPrototypes](#) rather than [PortInterfaces](#). Therefore, most communication-relevant attributes are related to the [PortPrototypes](#) of an [SwComponentType](#).

The communication attributes are organized in a so-called **communication specification** (in terms of the meta-model: [ComSpec](#)) classes.]()

Note that the communication specification is optional, i.e. its existence is not required in any case. Figures 4.29 and 4.30 provide an overview of communication specifications. The derived meta-classes are explained in the following sub-chapters.

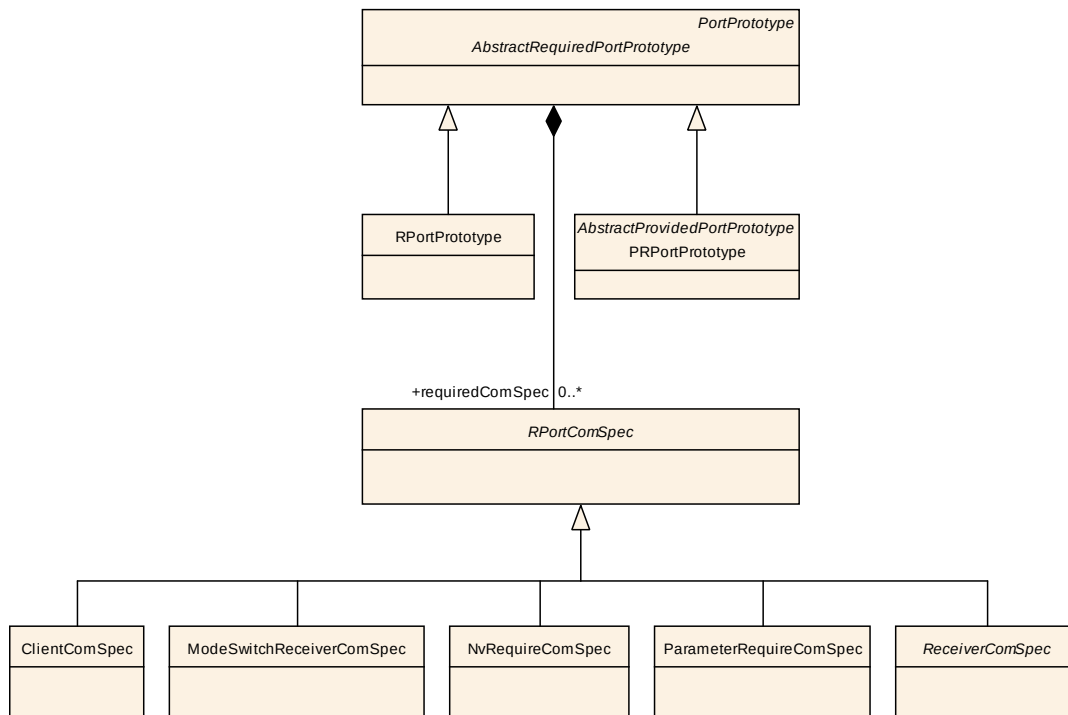


Figure 4.29: Overview of communication attributes of [RPortPrototype](#)

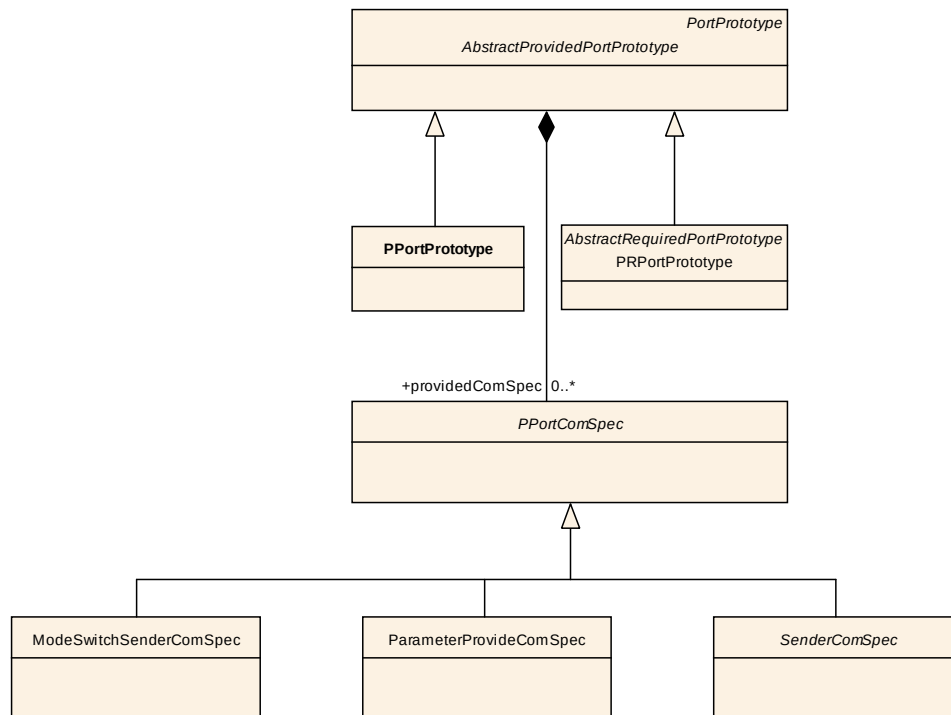


Figure 4.30: Overview of communication attributes of `PPortPrototype`

As explained before, `ComSpec` meta-classes which are required on the level of a `SwComponentType` are attached to the `PortPrototype` declarations which in turn are part of the definition of a `SwComponentType`. Nevertheless, the usage of `ComSpecs` is **not** restricted to the `PortPrototypes` of `AtomicSwComponentTypes` (for more details please refer to section 2.5).

Sections 7.5.1 and 7.5.2 then explain the sender-receiver and client-server communication patterns with respect to the RTE, the RTE events and the corresponding communication attributes.

Several `ComSpecs` allow to define `initValues` in relation to the associated `DataPrototype`. For further details about the representation of `initValues` please refer to section 5.7.2.

Furthermore, semantic constraints apply such that specific subclasses of `ComSpec` can only be owned by `PortPrototypes` typed by the corresponding kind of `PortInterface`.

[constr_1290] Limitation on the number of `PPortComSpecs` in the context of one `PPortPrototype` [Within the context of one `PPortPrototype` there can only be **one** `PPortComSpec` that references a given `dataElement` or `clientServerOperation`.]()

In other words, it is not allowed that two or more `PPortComSpec` exist in the context of a one `PPortPrototype` that refer to the same `dataElement` or `clientServerOperation`.

[constr_1291] Limitation on the number of RPortComSpecs in the context of one PPortPrototype [Within the context of one RPortPrototype, there can only be one RPortComSpec that references a given dataElement or clientServerOperation.]()

In other words, it is not allowed that two or more RPortComSpec exist in the context of a one RPortPrototype that refer to the same dataElement or clientServerOperation.

[TPS_SWCT_01454] PRPortPrototype can own both RPortComSpecs and PPortComSpecs [In contrast to PPortPrototype and RPortPrototype, PRPortPrototype can own both RPortComSpecs and PPortComSpecs at the same time.](RS_SWCT_03250)

Nevertheless, the following restriction applies:

[constr_1292] Limitation on the number of RPortComSpecs/PPortComSpecs in the context of one PRPortPrototype [Within the context of one PRPortPrototype, there can only be one RPortComSpec and one PPortComSpec that references a given dataElement or clientServerOperation.]()

In other words, it is not allowed that two or more PPortComSpec exist in the context of a one PRPortPrototype that refer to the same dataElement or clientServerOperation. In the same manner, not allowed that two or more RPortComSpec exist in the context of a one PRPortPrototype that refer to the same dataElement or clientServerOperation.

The rationale for the existence of [constr_1290], [constr_1291], and [constr_1292] is that the AUTOSAR communication layer needs an unambiguous specification of the communication behavior. The existence of redundant RPortComSpecs/PPortComSpecs may easily be contradicting each other and this would inhibit the creation of a valid configuration for the AUTOSAR Com.

Class	PPortComSpec (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
Note	Communication attributes of a provided PortPrototype. This class will contain attributes that are valid for all kinds of provide ports, independent of client-server or sender-receiver communication patterns.			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
—	—	—	—	—

Table 4.57: PPortComSpec

Class	RPortComSpec (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
Note	Communication attributes of a required PortPrototype. This class will contain attributes that are valid for all kinds of require-ports, independent of client-server or sender-receiver communication patterns.			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
–	–	–	–	–

Table 4.58: RPortComSpec

[constr_1043] PortInterface vs. ComSpec [The allowed combinations of a specific kind of [PortInterface](#) and a kind of [ComSpec](#) are documented in Table 4.59.]()

PortInterface	ComSpec
SenderReceiverInterface	SenderComSpec , ReceiverComSpec
ClientServerInterface	ClientComSpec , ServerComSpec
ModeSwitchInterface	ModeSwitchSenderComSpec , ModeSwitchReceiverComSpec
ParameterInterface	ParameterProvideComSpec , ParameterRequireComSpec
NvDataInterface	NvRequireComSpec , NvProvideComSpec

Table 4.59: [PortInterface](#) vs. [ComSpec](#)

As explained in section 2.5, there are cases where [PortPrototypes](#) owned by a [CompositionSwComponentType](#) could have `initValues`.

Therefore, it is possible that [PortPrototypes](#) owned by [CompositionSwComponentTypes](#) can have [ComSpecs](#). It is *not* required that the [ComSpecs](#) defined on the composition level match the [ComSpecs](#) defined inside the [CompositionSwComponentType](#).

If consistency would be required this constraint might be a major obstacle for integrating existing [AtomicSwComponentTypes](#) into a [CompositionSwComponentType](#) that has [PortPrototypes](#) with [ComSpecs](#).

4.5.1 Communication Specification for Sender-Receiver Communication

Communication specification applies in different ways to specific kinds of communication. Figure 4.31 shows the meta-model of the communication attributes relevant sender-receiver communication at an [RPortPrototype](#).

[TPS_SWCT_01455] Duplicate existence of `initValue` in the context of a [PR-PortPrototype](#) [If an `initValue` is defined in a [NonqueuedReceiverComSpec](#) owned by a [PRPortPrototype](#) its value shall be ignored.]([RS_SWCT_03250](#))

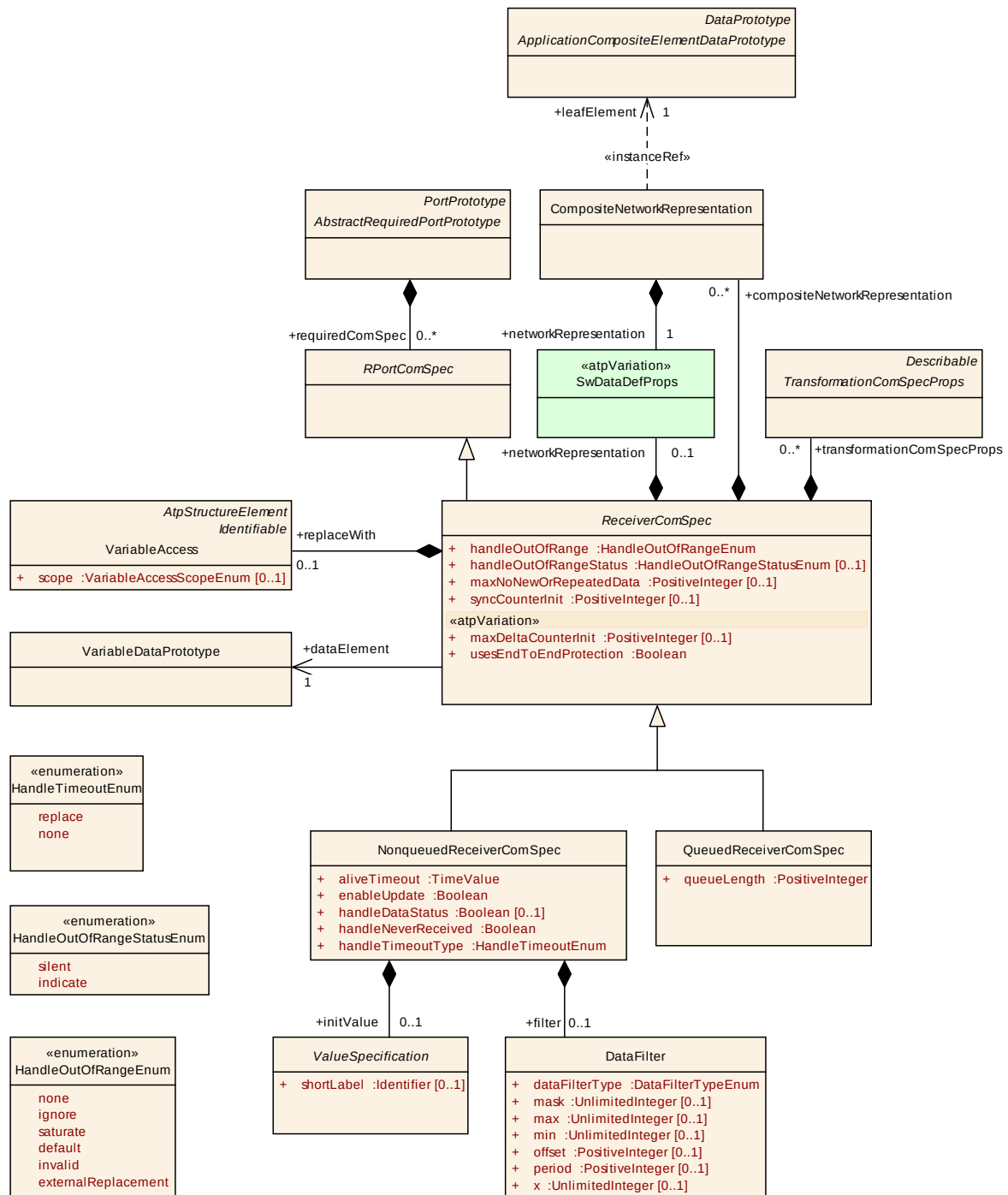


Figure 4.31: Communication attributes of *RPortPrototype* with respect to sender-receiver communication.

[TPS_SWCT_01219] ComSpec for queued and non-queued sender-receiver communication [Sender-receiver communication might be queued or non-queued. This aspect is primarily reflected in the value of `dataElement.swDataDefProps.swImplPolicy`. If the value of this attribute is set to `queued` then `QueuedSenderComSpec` and/or `QueuedReceiverComSpec` shall be defined. In all other applicable cases `NonqueuedSenderComSpec` resp. `NonqueuedReceiverComSpec` shall be used.

Thus, the constraints [constr_1129], [constr_1130], [constr_1131], and [constr_1132] shall apply.

While in the case of queued communication the `queueLength` attribute remains the only information item the non-queued case foresees several attributes for controlling communication behavior. `⌋()`

Class	ReceiverComSpec (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
Note	Receiver-specific communication attributes (RPortPrototype typed by SenderReceiverInterface).			
Base	ARObject, RPortComSpec			
Attribute	Datatype	Mul.	Kind	Note
compositeNetworkRepresentation	CompositeNetworkRepresentation	*	aggr	This represents a CompositeNetworkRepresentation defined in the context of a ReceiverComSpec. The purpose of this aggregation is to be able to specify the network representation of leaf elements of ApplicationCompositeDataTypes.
dataElement	VariableDataPrototype	1	ref	Data element these attributes belong to.
handleOutOfRange	HandleOutOfRangeEnum	1	attr	This attribute controls how values that are out of the specified range are handled according to the values of HandleOutOfRangeEnum.
handleOutOfRangeStatus	HandleOutOfRangeStatusEnum	0..1	attr	Control the way how return values are created in case of an out-of-range situation.
maxDeltaCounterInit	PositiveInteger	0..1	attr	Initial maximum allowed gap between two counter values of two consecutively received valid Data, i.e. how many subsequent lost data is accepted. For example, if the receiver gets Data with counter 1 and MaxDeltaCounterInit is 1, then at the next reception the receiver can accept Counters with values 2 and 3, but not 4. Note that if the receiver does not receive new Data at a consecutive read, then the receiver increments the tolerance by 1. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime
maxNumberOfRepeatedData	PositiveInteger	0..1	attr	The maximum amount of missing or repeated Data which the receiver does not expect to exceed under normal communication conditions.
networkRepresentation	SwDataDefinitions	0..1	aggr	A networkRepresentation is used to define how the dataElement is mapped to a communication bus.
replaceWith	VariableAccess	0..1	aggr	This aggregation is used to identify the AutosarDataPrototype to be taken for sourcing an external replacement in the out-of-range handling.

Attribute	Datatype	Mul.	Kind	Note
syncCounterInit	PositiveInteger	0..1	attr	Number of Data required for validating the consistency of the counter that shall be received with a valid counter (i.e. counter within the allowed lock-in range) after the detection of an unexpected behavior of a received counter.
transformationComSpecProps	TransformationComSpecProps	*	aggr	This references the TransformationComSpecProps which define port-specific configuration for data transformation.
usesEndToEndProtection	Boolean	1	attr	This indicates whether the corresponding dataElement shall be transmitted using end-to-end protection. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime

Table 4.60: ReceiverComSpec

Class	NonqueuedReceiverComSpec			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
Note	Communication attributes specific to non-queued receiving.			
Base	ARObject, RPortComSpec , ReceiverComSpec			
Attribute	Datatype	Mul.	Kind	Note
aliveTimeout	TimeValue	1	attr	Specify the amount of time (in seconds) after which the software component (via the RTE) needs to be notified if the corresponding data item have not been received according to the specified timing description. If the aliveTimeout attribute is 0 no timeout monitoring shall be performed.
enableUpdate	Boolean	1	attr	This attribute controls whether application code is entitled to check whether the value of the corresponding VariableDataPrototype has been updated.
filter	DataFilter	0..1	aggr	The applicable filter algorithm for filtering the value of the corresponding dataElement.
handleDataStatus	Boolean	0..1	attr	If this attribute is set to true than the Rte_IStatus API shall exist. If the attribute does not exist or is set to false then the Rte_IStatus API may still exist in response to the existence of further conditions.

Attribute	Datatype	Mul.	Kind	Note
handleNeverReceived	Boolean	1	attr	This attribute specifies whether for the corresponding VariableDataPrototype the "never received" flag is available. If yes, the RTE is supposed to assume that initially the VariableDataPrototype has not been received before. After the first reception of the corresponding VariableDataPrototype the flag is cleared. - If the value of this attribute is set to "true" the flag is required. - If set to "false", the RTE shall not support the "never received" functionality for the corresponding VariableDataPrototype.
handleTimeoutType	HandleTimeoutEnum	1	attr	This attribute controls the behavior with respect to the handling of timeouts.
initValue	ValueSpecification	0..1	aggr	Initial value to be used in case the sending component is not yet initialized. If the sender also specifies an initial value the receiver's value will be used.

Table 4.61: NonqueuedReceiverComSpec

Class	QueuedReceiverComSpec			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
Note	Communication attributes specific to queued receiving.			
Base	ARObject, RPortComSpec, ReceiverComSpec			
Attribute	Datatype	Mul.	Kind	Note
queueLength	PositiveInteger	1	attr	Length of queue for received events.

Table 4.62: QueuedReceiverComSpec

Enumeration	HandleTimeoutEnum
Package	M2::AUTOSARTemplates::SWComponentTemplate::Communication
Note	Strategies of handling a reception timeout violation.
Literal	Description
none	If set to none no replacement shall take place.
replace	If set to replace, the replacement value used shall be the ComInitValue.

Table 4.63: HandleTimeoutEnum

Primitive	TimeValue
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::PrimitiveTypes

Note	This primitive type is taken for expressing time values. The numerical value is supposed to be interpreted in the physical unit second. Tags: xml.xsd.customType=TIME-VALUE; xml.xsd.type=double

Table 4.64: TimeValue

[constr_1103] NonqueuedReceiverComSpec and enableUpdate [A `NonqueuedReceiverComSpec` that has attribute `enableUpdate` set to `true` may not reference a `dataElement` that in turn is referenced by a `VariableAccess` in the role `dataReadAccess`.]()

[constr_1201] initValue shall exist in an RPortPrototype [The optional attribute `initValue` **shall exist** if the enclosing `NonqueuedReceiverComSpec` is owned by an `RPortPrototype`.]()

[constr_1129] swImplPolicy and NonqueuedReceiverComSpec [The attribute `swImplPolicy` of a `dataElement` referenced by a `NonqueuedReceiverComSpec` **shall not** be set to the value `queued`.]()

[constr_1130] swImplPolicy and QueuedReceiverComSpec [The attribute `swImplPolicy` of a `dataElement` referenced by a `QueuedReceiverComSpec` **shall** be set to the value `queued`.]()

[constr_1188] Existence of ReceiverComSpec.replaceWith [The aggregation of `VariableAccess` in the role `ReceiverComSpec.replaceWith` shall exist **if and only if at least one of the following conditions is fulfilled**:

- Attribute `ReceiverComSpec.handleOutOfRange` is set to the value `externalReplacement`.
- Attribute `SenderReceiverInterface.invalidationPolicy.handleInvalid` is set to the value `externalReplacement`.

]()

[constr_1131] swImplPolicy and NonqueuedSenderComSpec [The attribute `swImplPolicy` of a `dataElement` referenced by a `NonqueuedSenderComSpec` **shall not** be set to the value `queued`.]()

[constr_1132] swImplPolicy and QueuedSenderComSpec [The attribute `swImplPolicy` of a `dataElement` referenced by a `QueuedSenderComSpec` **shall** be set to the value `queued`.]()

[TPS_SWCT_01220] initValue defines an initial value that shall be taken if the corresponding dataElement has not yet been received [The aggregation of `ValueSpecification` in the role `initValue` defines an initial value that shall be taken if the corresponding `dataElement` has not yet been received but the application software is attempting to access its value.

This is the only relevant definition of an initial value for data transmission. That is, any `initValue` defined in the context of `VariableDataPrototype` is ignored! `]()`

The communication attributes on the sender side are sketched in Figure 4.33.

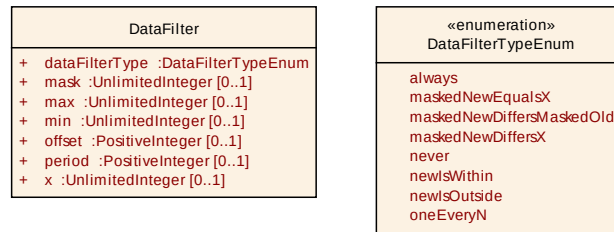


Figure 4.32: DataFilter and its communication attributes.

Figure 4.32 shows the model of the communication attributes relevant for defining data filters.

[TPS_SWCT_01221] DataFilter `[` For every `RPortPrototype` typed by a `SenderReceiverInterface` a `DataFilter` can be defined given that non-queued communication is foreseen. `]()`

Fifteen filter algorithms formally described by the enumeration type `DataFilterTypeEnum` in the meta-model are taken from OSEK COM 3.0.3 specification [18] that is referenced by the RTE specification [2].

[TPS_SWCT_01222] Applicability of DataFilter `[` This OSEK specification states that “filtering is only used for messages that can be interpreted as C language unsigned integer types (characters, unsigned integers and enumerations).” `](RS_SWCT_03221)`

[constr_1044] Applicability of DataFilter `[` According to the origin of `DataFilter`, i.e. OSEK COM 3.0.3 specification [18], `DataFilters` can only be applied to values with an integer base type. `]()`

Class	DataFilter			
Package	M2::AUTOSARTemplates::CommonStructure::Filter			
Note	Base class for data filters. The type of the filter is specified in attribute <code>dataFilterType</code> . Some of the filter types require additional arguments which are specified as attributes of this class.			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
<code>dataFilterType</code>	<code>DataFilterTypeEnum</code>	1	attr	This attribute specifies the type of the filter.
<code>mask</code>	<code>UnlimitedInteger</code>	0..1	attr	Mask for old and new value.
<code>max</code>	<code>UnlimitedInteger</code>	0..1	attr	Value to specify the upper boundary
<code>min</code>	<code>UnlimitedInteger</code>	0..1	attr	Value to specify the lower boundary
<code>offset</code>	<code>PositiveInteger</code>	0..1	attr	Specifies the initial number of messages to occur before the first message is passed
<code>period</code>	<code>PositiveInteger</code>	0..1	attr	Specifies number of messages to occur before the message is passed again
<code>x</code>	<code>UnlimitedInteger</code>	0..1	attr	Value to compare with

Attribute	Datatype	Mul.	Kind	Note

Table 4.65: DataFilter

Enumeration	DataFilterTypeEnum
Package	M2::AUTOSARTemplates::CommonStructure::Filter
Note	This enum specifies the supported DataFilterTypes.
Literal	Description
always	No filtering is performed so that the message always passes.
masked NewDiffers MaskedOld	Pass messages where the masked value has changed. (new_value&mask) !=(old_value&mask) new_value: current value of the message old_value: last value of the message (initialized with the initial value of the message, updated with new_value if the new message value is not filtered out)
maskedNew DiffersX	Pass messages whose masked value is not equal to a specific value x (new_value&mask) != x new_value: current value of the message
maskedNew EqualsX	Pass messages whose masked value is equal to a specific value x (new_value&mask) == x new_value: current value of the message
never	The filter removes all messages.
newIsOutside	Pass a message if its value is outside a predefined boundary. (min > new_value) OR (new_value > max)
newIsWithin	Pass a message if its value is within a predefined boundary. min <= new_value <= max
oneEveryN	Pass a message once every N message occurrences. Algorithm: occurrence % period == offset Start: occurrence = 0. Each time the message is received or transmitted, occurrence is incremented by 1 after filtering. Length of occurrence is 8 bit (minimum).

Table 4.66: DataFilterTypeEnum

[TPS_SWCT_01593] Semantics of attribute `ReceiverComSpec.transformationComSpecProps` [The `ReceiverComSpec.transformationComSpecProps` is used to configure `PortPrototype`-specific properties for data transformation in case of receiving inter-ECU communication.]()

[constr_1323] Applicability of attribute `ReceiverComSpec.usesEndToEndProtection` [The attribute `ReceiverComSpec.usesEndToEndProtection` shall be set to `false` for all `ReceiverComSpec` that aggregate `EndToEndTransformationDescription` in the role `transformationComSpecProps`.]()

See chapter 4.5.6 for details.

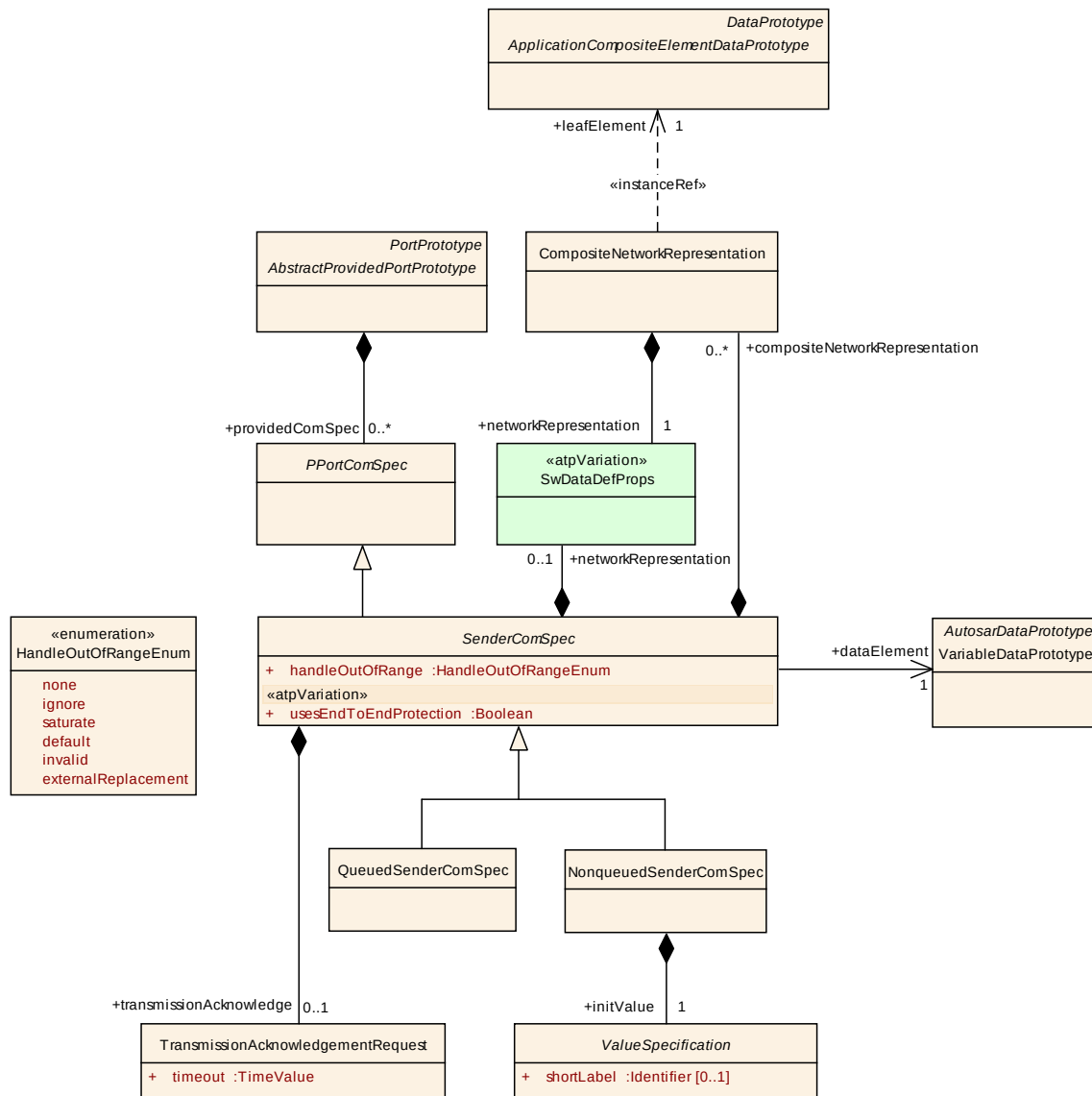


Figure 4.33: Communication attributes of *PPortPrototype* with respect to sender-receiver communication.

Class	SenderComSpec (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
Note	Communication attributes for a sender port (PPortPrototype typed by SenderReceiverInterface).			
Base	ARObject, PPortComSpec			
Attribute	Datatype	Mul.	Kind	Note
compositeNetworkRepresentation	CompositeNetworkRepresentation	*	aggr	This represents a CompositeNetworkRepresentation defined in the context of a SenderComSpec.
dataElement	VariableDataPrototype	1	ref	Data element these quality of service attributes apply to.
handleOutOfRange	HandleOutOfRangeEnum	1	attr	This attribute controls how out-of-range values shall be dealt with.

Attribute	Datatype	Mul.	Kind	Note
networkRepresentation	SwDataDefinitions	0..1	aggr	A networkRepresentation is used to define how the dataElement is mapped to a communication bus.
transmissionAcknowledgement	TransmissionAcknowledgementRequest	0..1	aggr	Requested transmission acknowledgement for data element.
usesEndToEndProtection	Boolean	1	attr	This indicates whether the corresponding dataElement shall be transmitted using end-to-end protection. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime

Table 4.67: SenderComSpec

Class	QueuedSenderComSpec			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
Note	Communication attributes specific to distribution of events (PPortPrototype, SenderReceiverInterface and dataElement carries an "event").			
Base	ARObject, PPortComSpec, SenderComSpec			
Attribute	Datatype	Mul.	Kind	Note
—	—	—	—	—

Table 4.68: QueuedSenderComSpec

Class	NonqueuedSenderComSpec			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
Note	Communication attributes for non-queued sender/receiver communication (sender side)			
Base	ARObject, PPortComSpec, SenderComSpec			
Attribute	Datatype	Mul.	Kind	Note
initValue	ValueSpecification	1	aggr	Initial value to be sent if sender component is not yet fully initialized, but receiver needs data already.

Table 4.69: NonqueuedSenderComSpec

Class	TransmissionAcknowledgementRequest			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
Note	Requests transmission acknowledgement that data has been sent successfully. Success/failure is reported via a SendPoint of a RunnableEntity.			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
timeout	TimeValue	1	attr	Number of seconds before an error is reported or in case of allowed redundancy, the value is sent again.

Table 4.70: TransmissionAcknowledgementRequest

<i>Enumeration</i>	HandleOutOfRangeEnum
Package	M2::AUTOSARTemplates::SWComponentTemplate::Communication
Note	A value of this type is taken for controlling the range checking behavior of the AUTOSAR RTE.
Literal	Description
default	The RTE will use the initValue if the actual value is out of the specified bounds.
external Replacement	This indicates that the value replacement is sourced from the attribute replaceWith.
ignore	The RTE will ignore any attempt to send or receive the corresponding dataElement if the value is out of the specified range.
invalid	The RTE will use the invalidValue if the value is out of the specified bounds.
none	A range check is not required.
saturate	The RTE will saturate the value of the dataElement such that it is limited to the applicable upper bound if it is greater than the upper bound. Consequently, it is limited to the applicable lower bound if the value is less than the lower bound.

Table 4.71: HandleOutOfRangeEnum

[TPS_SWCT_01223] **networkRepresentation** defines how a specific **dataElement** is represented on a communication bus [For sender-receiver communication, it is possible to specify how **dataElements** are represented given that the communication requires the usage of a dedicated communication bus.

That is, by means of the **networkRepresentation** it is possible to define how a specific **dataElement** is represented on a communication bus. For this purpose the **networkRepresentation** is implemented as an aggregation of **SwDataDefProps**.
]()

[TPS_SWCT_01224] **CompuMethods** of **dataElement** and the **networkRepresentation** are used for conversion purposes [The attached **CompuMethods** of both the **dataElement** and the **networkRepresentation** can be used to identify the conversion between the two. The advantage of this approach is that this can also be used without any modifications in combination with a general remapping and rescaling of **dataElements** between different **SwComponentTypes**, regardless whether they are located on the same or on different ECUs.]()

Please note that the decision whether or not to take the **networkRepresentation** for data mapping is done in the context of the AUTOSAR System Template [11]. Please find more detailed information about this aspect in the applicable specification.

[TPS_SWCT_01452] **Applicability of networkRepresentation for ApplicationCompositeDataType** [The aggregation of **networkRepresentation** at the **ReceiverComSpec** or **SenderComSpec** only applies for **dataElements** typed by **ApplicationPrimitiveDataTypes**. For the case of using an **ApplicationCompositeDataType** an additional mechanism shall be used.

In particular, **compositeNetworkRepresentation** shall be used to define the **networkRepresentation** of **leaf elements** of **ApplicationCompositeDataTypes**.
]()

[constr_1196] Existence of `networkRepresentation` vs. `compositeNetworkRepresentation` [If a `ReceiverComSpec` or `SenderComSpec` aggregates `networkRepresentation` it shall **not** aggregate `compositeNetworkRepresentation` at the same time (and vice versa).]()

[constr_1197] Existence of `compositeNetworkRepresentation` shall be comprehensive [If at least one `compositeNetworkRepresentation` exists then for each leaf `ApplicationCompositeElementDataPrototype` of the affected `ApplicationCompositeDataType` exactly one `compositeNetworkRepresentation` shall be defined.]()

Granted, the definition of **[constr_1197]** to some extent has a recursive character. The meaning is that if it is actually intended to define a `compositeNetworkRepresentation` then the definition shall be completely covering the entire set of leaf elements of the corresponding `ApplicationCompositeDataType`. In other words, it's all or nothing.

Class	CompositeNetworkRepresentation			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
Note	This meta-class is used to define the network representation of leaf elements of composite application data types.			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
leafElement	<code>ApplicationCompositeElementDataPrototype</code>	1	iref	This represents that leaf element of an application composite data type.
networkRepresentation	<code>SwDataDefProps</code>	1	aggr	The <code>SwDataDefProps</code> owned by the <code>CompositeNetworkRepresentation</code> are used to define the network representation of the leaf element of an <code>ApplicationCompositeDataType</code> .

Table 4.72: CompositeNetworkRepresentation

4.5.2 Communication Specification for Client-Server Communication

The communication aspects relevant for client communication are sketched in Figure 4.34.

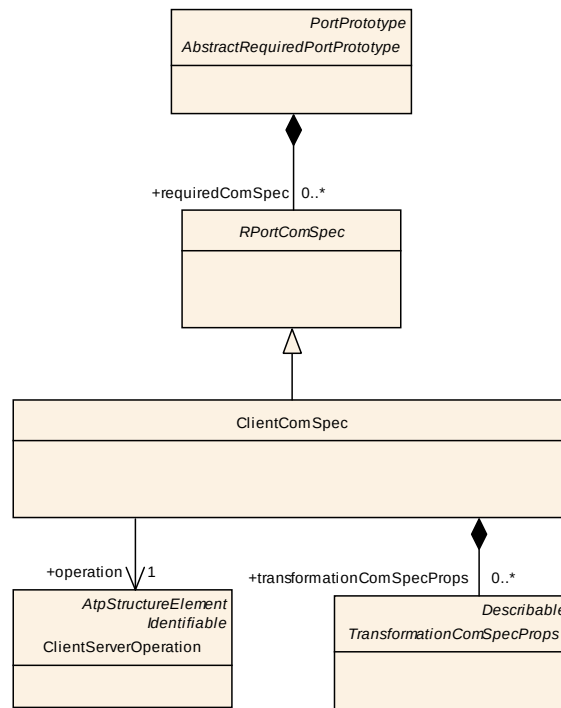


Figure 4.34: Communication attributes of *RPortPrototype* with respect to client-server communication.

Class	ClientComSpec			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
Note	Client-specific communication attributes (RPortPrototype typed by ClientServerInterface).			
Base	ARObject, RPortComSpec			
Attribute	Datatype	Mul.	Kind	Note
operation	ClientServerOperation	1	ref	This represents the corresponding ClientServerOperation.
transformationComSpecProps	TransformationComSpecProps	*	aggr	This references the TransformationComSpecProps which define port-specific configuration for data transformation.

Table 4.73: ClientComSpec

The server side looks very similar but provides an attribute for specifying the queue length.

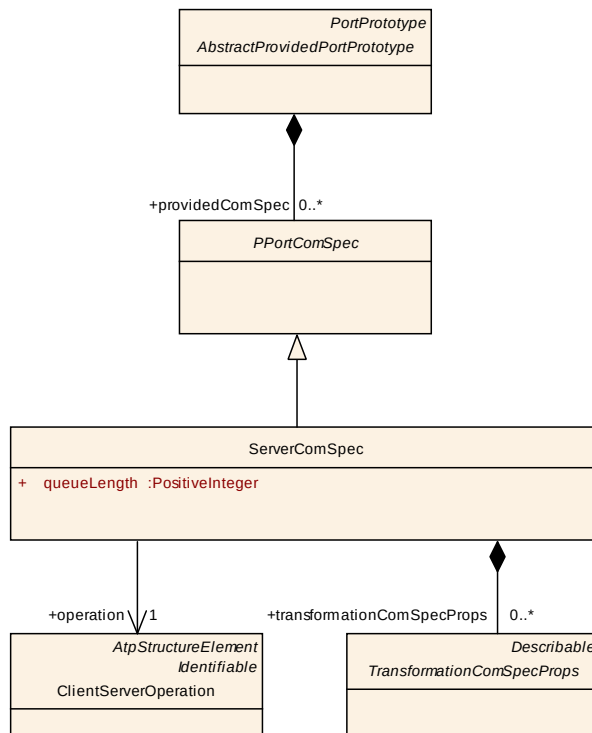


Figure 4.35: Communication attributes of [PPortPrototype](#) with respect to client-server communication.

Class	ServerComSpec			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
Note	Communication attributes for a server port (PPortPrototype and ClientServerInterface).			
Base	ARObject, PPortComSpec			
Attribute	Datatype	Mul.	Kind	Note
operation	ClientServerOperation	1	ref	Operation these communication attributes apply to.
queueLength	PositiveInteger	1	attr	Length of call queue on the server side. The queue is implemented by the RTE. The value shall be greater or equal to 1. Setting the value of queueLength to 1 implies that incoming requests are rejected while another request that arrived earlier is being processed.
transformationComSpecProps	TransformationComSpecProps	*	aggr	This references the TransformationComSpecProps which define port-specific configuration for data transformation.

Table 4.74: ServerComSpec

[TPS_SWCT_01225] [RunnableEntity](#) implements the functionality of two or more [ClientServerOperations](#) [Please note that it is technically possible to let a single [RunnableEntity](#) implement the functionality of two or more [ClientServerOperations](#). For this purpose two or more [OperationInvokedEvents](#) need to reference this single [RunnableEntity](#).

In this case, however, it is essential that the queue length associated with each of the `ClientServerOperations` has the same value. In other words: `]()`

[constr_1128] Queue length of `ClientServerOperations` associated with the same `RunnableEntity` `] If two or more OperationInvokedEvents reference a single RunnableEntity the value of the ServerComSpec attribute queueLength shall be identical for all ServerComSpecs owned by PPortPrototypes of the enclosing SwComponentType that reference one of the ClientServerOperations that are also referenced by the OperationInvokedEvents.]()`

[TPS_SWCT_01595] Semantics of attribute `ClientComSpec.transformationComSpecProps` `] The attribute ClientComSpec.transformationComSpecProps shall be used to configure PortPrototype-specific properties for data transformation in case of Client/Server inter-ECU communication for the reception of the server's response. ](RS_SWCT_03221)`

[TPS_SWCT_01596] Semantics of attribute `ServerComSpec.transformationComSpecProps` `] The attribute ServerComSpec.transformationComSpecProps shall be used to configure PortPrototype-specific properties for data transformation in case of Client/Server inter-ECU communication for the reception of the client's request. ](RS_SWCT_03221)`

See chapter 4.5.6 for details.

4.5.3 Communication Specification for Mode Switch Communication

In analogy to the previous section, Figure 4.36 shows the meta-model elements relevant for a mode switch communication. On the sender side it is possible to specify that an acknowledgement is supposed to be returned that indicates the successful processing of the mode switch request.

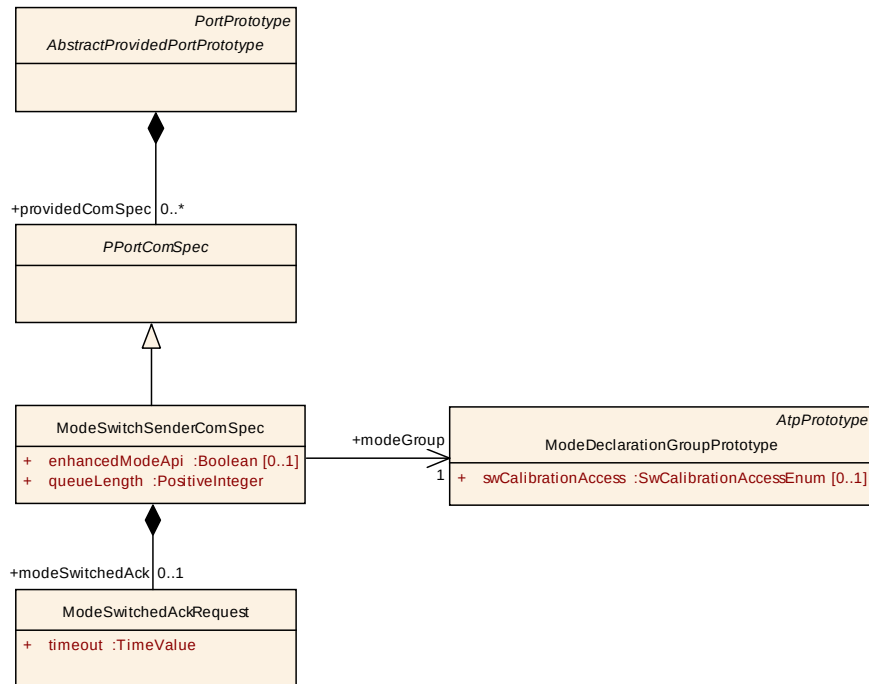


Figure 4.36: Communication attributes of **PPortPrototype with respect to mode switch communication.**

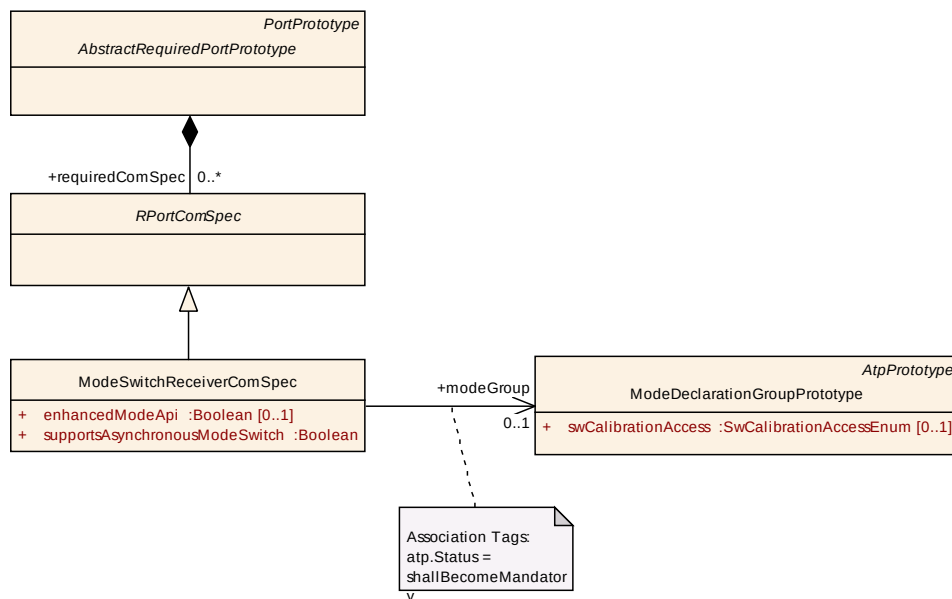


Figure 4.37: Communication attributes of **PPortPrototype with respect to mode switch communication.**

[TPS_SWCT_01514] Duplicate existence of **enhancedModeApi** in the context of a **PRPortPrototype** [If the attribute **enhancedModeApi** is defined in a **ModeSwitchReceiverComSpec** owned by a **PRPortPrototype** its value shall be ignored.] (**RS_SWCT_03250**)

Class	ModeSwitchSenderComSpec			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
Note	Communication attributes of PPortPrototypes with respect to mode communication			
Base	ARObject, PPortComSpec			
Attribute	Datatype	Mul.	Kind	Note
enhanced ModeApi	Boolean	0..1	attr	This controls the creation of the enhanced mode API that returns information about the previous mode and the next mode. If set to "true" the enhanced mode API is supposed to be generated. For more details please refer to the SWS_RTE.
modeGroup	ModeDeclarationGroupPrototype	1	ref	ModeDeclarationGroupPrototype (of the same PortInterface) to which these communication attributes apply.
modeSwitchedAck	ModeSwitchedAckRequest	0..1	aggr	If this aggregation exists an acknowledgement for the successful processing of the mode switch request is required.
queueLength	PositiveInteger	1	attr	Length of call queue on the mode user side. The queue is implemented by the RTE. The value shall be greater or equal to 1. Setting the value of queueLength to 1 implies that incoming requests are rejected while another request that arrived earlier is being processed.

Table 4.75: ModeSwitchSenderComSpec

Class	ModeSwitchedAckRequest			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
Note	Requests acknowledgements that a mode switch has been proceeded successfully			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
timeout	TimeValue	1	attr	Number of seconds before an error is reported or in case of allowed redundancy, the value is sent again.

Table 4.76: ModeSwitchedAckRequest

Class	ModeSwitchReceiverComSpec			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
Note	Communication attributes of RPortPrototypes with respect to mode communication			
Base	ARObject, RPortComSpec			
Attribute	Datatype	Mul.	Kind	Note
enhanced ModeApi	Boolean	0..1	attr	This controls the creation of the enhanced mode API that returns information about the previous mode and the next mode. If set to "true" the enhanced mode API is supposed to be generated. For more details please refer to the SWS_RTE.

Attribute	Datatype	Mul.	Kind	Note
modeGroup	ModeDeclarationGroupPrototype	0..1	ref	ModeDeclarationGroupPrototype (of the same PortInterface) to which these communication attributes apply. Tags: atp.Status=shallBecomeMandatory
supportsAsynchronousModeSwitch	Boolean	1	attr	This attribute controls the behavior of the corresponding RPortPrototype with respect to the question whether it can deal with asynchronous mode switch requests, i.e. if set to true, the RPortPrototype is able to deal with an asynchronous mode switch request.

Table 4.77: ModeSwitchReceiverComSpec

4.5.4 Communication Specification for Parameters

Granted, the definition of a ComSpec for [ParameterDataPrototype](#)s looks strange on first sight. A [ParameterDataPrototype](#) owned by a [PPortPrototype](#) typed by a [ParameterInterface](#) is not actually transmitted over any communication medium. Therefore, the term *communication* should in this case be taken with a grain of salt.

However, it is generally necessary to be able to define role-specific initial values for [ParameterDataPrototype](#)s aggregated in a [ParameterInterface](#). In other words, the actual problem closely resembles the definition of initial values in the case of sender-receiver communication.

[TPS_SWCT_01226] initValue on the level of a ComSpec is relevant for connections to the corresponding PortPrototype [Please note that (along the example of sender-receiver communication) only the `initValue` defined in the context of a [ParameterProvideComSpec](#) or [ParameterRequireComSpec](#) is relevant for connections to the corresponding [PortPrototype](#). An `initValue` defined in the scope of a [ParameterDataPrototype](#) is ignored.]()

Therefore, it is only reasonable to apply the existing and well-known pattern to the definition of initial values for [ParameterDataPrototype](#)s aggregated in a [ParameterInterface](#). The actual modeling is sketched in Figure 4.38 for provided [ParameterDataPrototype](#)s and in Figure 4.39 for required [ParameterDataPrototype](#)s.

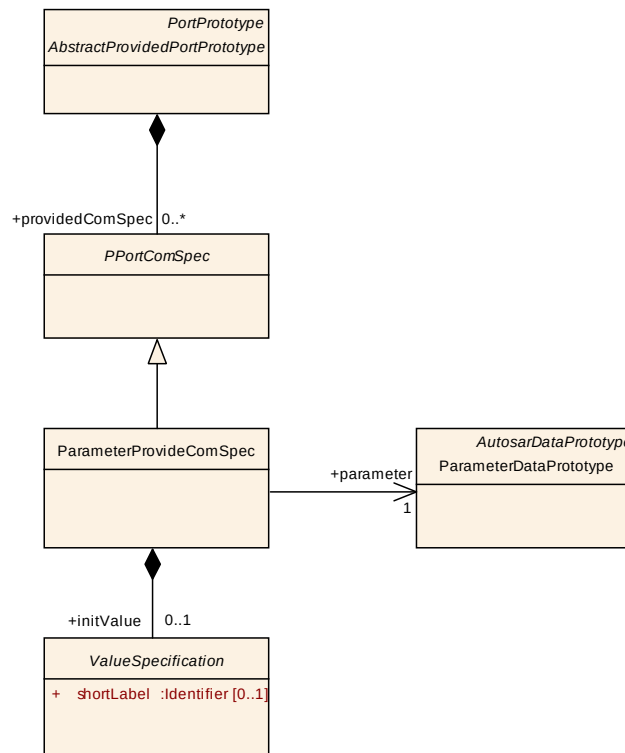


Figure 4.38: Communication attributes of [ParameterDataPrototypes](#) with respect to [PPortPrototype](#)

Class	ParameterProvideComSpec			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
Note	"Communication" specification that applies to parameters on the provided side of a connection.			
Base	ARObject, PPortComSpec			
Attribute	Datatype	Mul.	Kind	Note
initValue	ValueSpecification	0..1	aggr	The initial value applicable for the corresponding ParameterDataPrototype.
parameter	ParameterDataPrototype	1	ref	The ParameterDataPrototype to which the ParameterComSpec applies.

Table 4.78: ParameterProvideComSpec

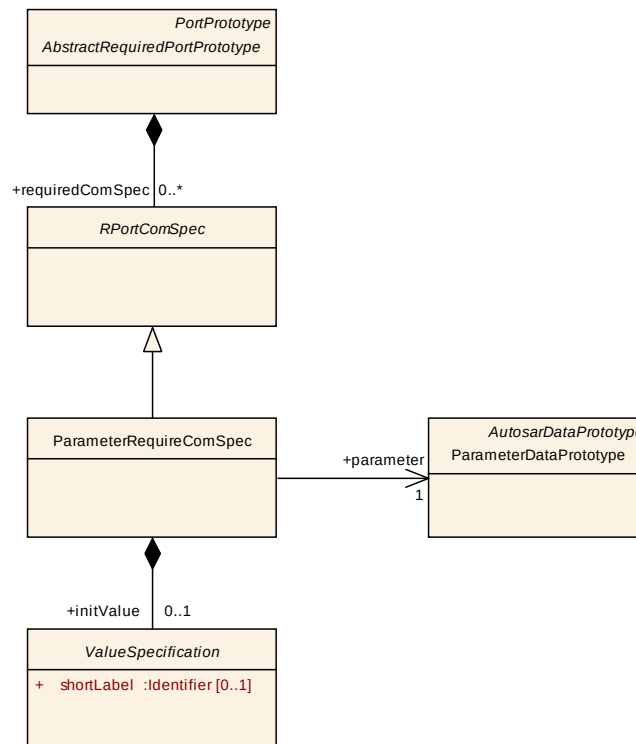


Figure 4.39: Communication attributes of [ParameterDataPrototypes](#) with respect to [RPortPrototype](#)

Class	ParameterRequireComSpec			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
Note	"Communication" specification that applies to parameters on the required side of a connection.			
Base	ARObject, RPortComSpec			
Attribute	Datatype	Mul.	Kind	Note
initValue	ValueSpecification	0..1	aggr	The initial value applicable for the corresponding ParameterDataPrototype .
parameter	ParameterDataPrototype	1	ref	The ParameterDataPrototype to which the ParameterRequireComSpec applies.

Table 4.79: ParameterRequireComSpec

4.5.5 Communication Specification for NV Data

[TPS_SWCT_01141] [AtomicSwComponentType](#) may have [AbstractRequiredPortPrototypes](#) typed by an [NvDataInterface](#) [An [AtomicSwComponentType](#) may have [AbstractRequiredPortPrototypes](#) typed by an [NvDataInterface](#). If such an [AbstractRequiredPortPrototype](#) remains unconnected the [nvData](#) still need to have reasonable value⁶.]([RS_SWCT_03225](#))

⁶Note that it is assumed that only a subset of meta-classes that inherit from [AtomicSwComponentType](#) will actually apply for the definition of initial values for [nvData](#). Most likely the [Application-](#)

[TPS_SWCT_01227] Unconnected **AbstractRequiredPortPrototype** typed by **NvDataInterface** [For this purpose it is possible to let the **AbstractRequiredPortPrototype** own an **NvRequireComSpec** that in turn owns a **ValueSpecification** in the role of **initValue**.

It is therefore possible to provide an **nvData** with a reasonable value even if the corresponding **AbstractRequiredPortPrototype** remains unconnected.](RS_SWCT_03225)

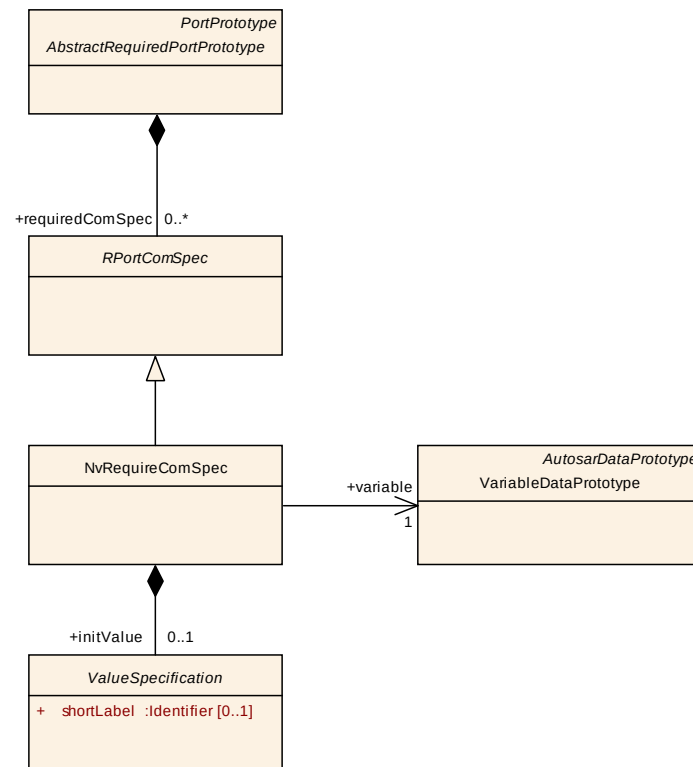


Figure 4.40: Communication attributes of a required **VariableDataPrototype used in the context of an **NvDataInterface****

Please note that (along the example of sender-receiver communication, see [TPS_SWCT_01226]) only the **initValue** defined in the context of a **NvRequireComSpec** is relevant for connections to the corresponding **PortPrototype**. An **initValue** defined in the scope of a **VariableDataPrototype** is ignored.

Class	NvRequireComSpec			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
Note	Communication attributes of RPortPrototypes with respect to Nv data communication on the required side.			
Base	ARObject, RPortComSpec			
Attribute	Datatype	Mul.	Kind	Note
initValue	ValueSpecification	0..1	aggr	The initial value owned by the NvComSpec

SwComponentType and the **SensorActuatorSwComponentType** will be candidates for using this feature but it will obviously not be reasonable for e.g. **NvBlockSwComponentType**.

Attribute	Datatype	Mul.	Kind	Note
variable	VariableDataPrototype	1	ref	The VariableDataPrototype the ComSpec applies for.

Table 4.80: NvRequireComSpec

[TPS_SWCT_01228] **NvProvideComSpec** [As communication with an **NvBlockSwComponentType** is in most cases bi-directional it is also necessary to consider role-specific communication attributes for **AbstractProvidedPortPrototypes** typed by an **NvDataInterface**. For this purpose the **NvProvideComSpec** (see Figure 4.41) is defined.

The main purpose of this kind of **ComSpec** is the definition of initial values for the **RAM Block** and the **ROM Block** that corresponds to an **nvData** defined in the context of the **NvDataInterface** used to type the given **AbstractProvidedPortPrototype**.](**RS_SWCT_03225**)

Note that these initial values can be taken as an input for designing an **NvBlockSwComponentType**, in particular the **ramBlocks** and **romBlocks** of **NvBlockDescriptors** owned by the **NvBlockSwComponentType**. Further details are explained in Figure 11.6.

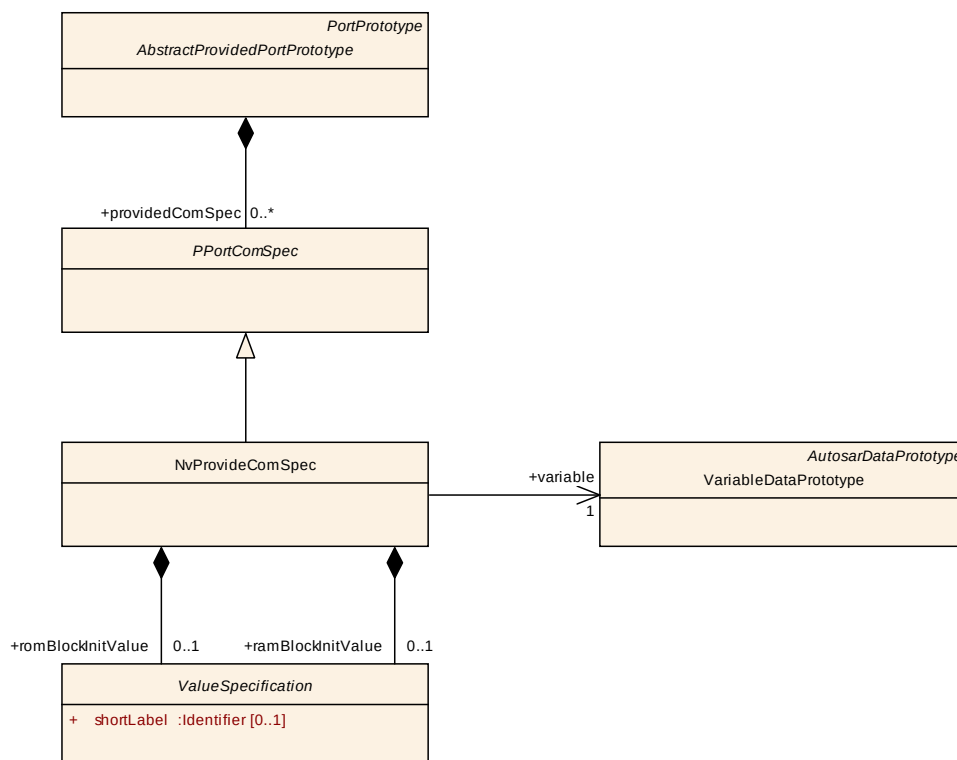


Figure 4.41: Communication attributes of a provided **VariableDataPrototypes used in the context of an **NvDataInterface****

In other words, by means of the **NvProvideComSpec** the author of an **ApplicationSwComponentType** can express detailed requirements on the later design of a corresponding **NvBlockSwComponentType**.

Class	NvProvideComSpec			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
Note	Communication attributes of PPortPrototypes with respect to Nv data communication on the provided side.			
Base	ARObject, PPortComSpec			
Attribute	Datatype	Mul.	Kind	Note
ramBlockInitValue	ValueSpecification	0..1	aggr	This represents the initial value of the RAM Block that corresponds to the referenced variable.
romBlockInitValue	ValueSpecification	0..1	aggr	This represents the initial value of the ROM block that corresponds to the referenced variable.
variable	VariableDataPrototype	1	ref	This represents the variable for which the ComSpec is specified.

Table 4.81: NvProvideComSpec

4.5.6 Configuration of Data Transformation

Using the [TransformationComSpecProps](#) it is possible to define configuration options for specific transformers of inter-ecu communication which is subject to data transformation.

[TPS_SWCT_01594] Semantics of [TransformationComSpecProps](#) [The definition of a [TransformationComSpecProps](#) can always be provided in the SWC description but the configuration shall **only** have an effect if

1. the actual communication involves at least two [EcuInstances](#)
2. the respective data transformer (given by the used [TransformationComSpecProps](#)) is used during data transformation (see [DataTransformation](#))

]([RS_SWCT_03221](#))

For clarification, the configuration given in [TransformationComSpecProps](#) will simply be ignored if the conditions defined by [\[TPS_SWCT_01594\]](#) do not apply.

[TPS_SWCT_01597] PortPrototype-specific data transformation configuration [Meta-class [TransformationComSpecProps](#) shall be used for the specification of [PortPrototype](#)-specific configuration options for data transformation of inter-ECU communication.]([RS_SWCT_03221](#))

Please note that only some transformers offer [PortPrototype](#)-specific configuration (e.g. SOME/IP transformer doesn't have [TransformationComSpecProps](#)).

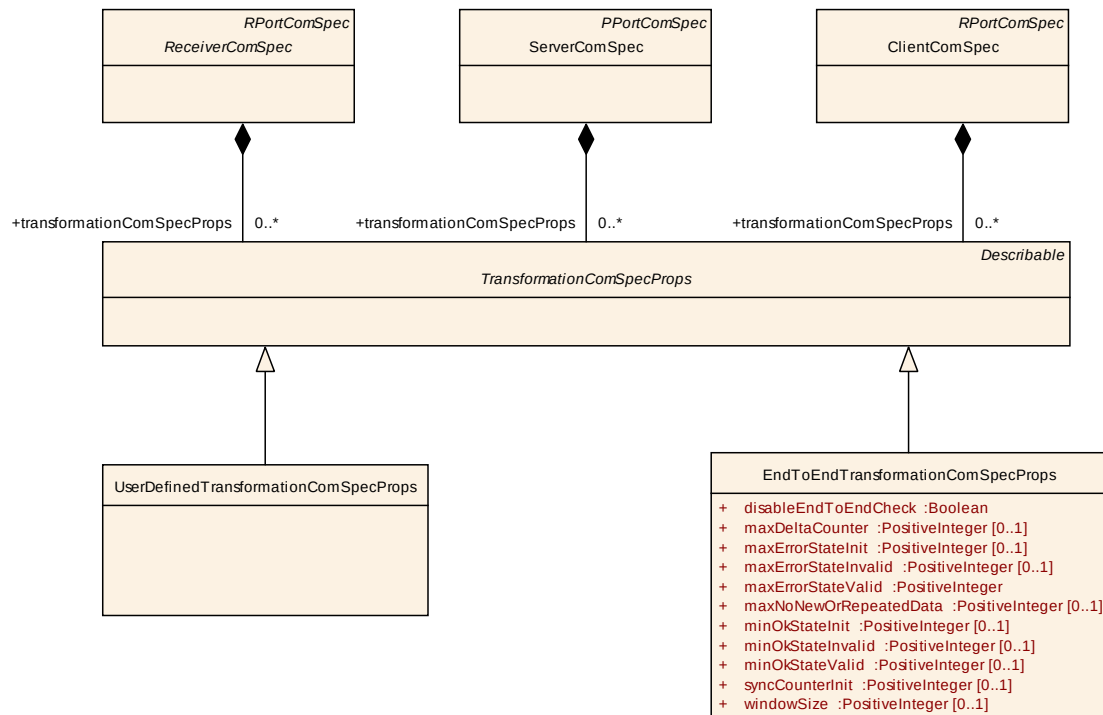


Figure 4.42: Specification of data transformation properties within [ReceiverComSpec](#), [ServerComSpec](#), and [ClientComSpec](#)

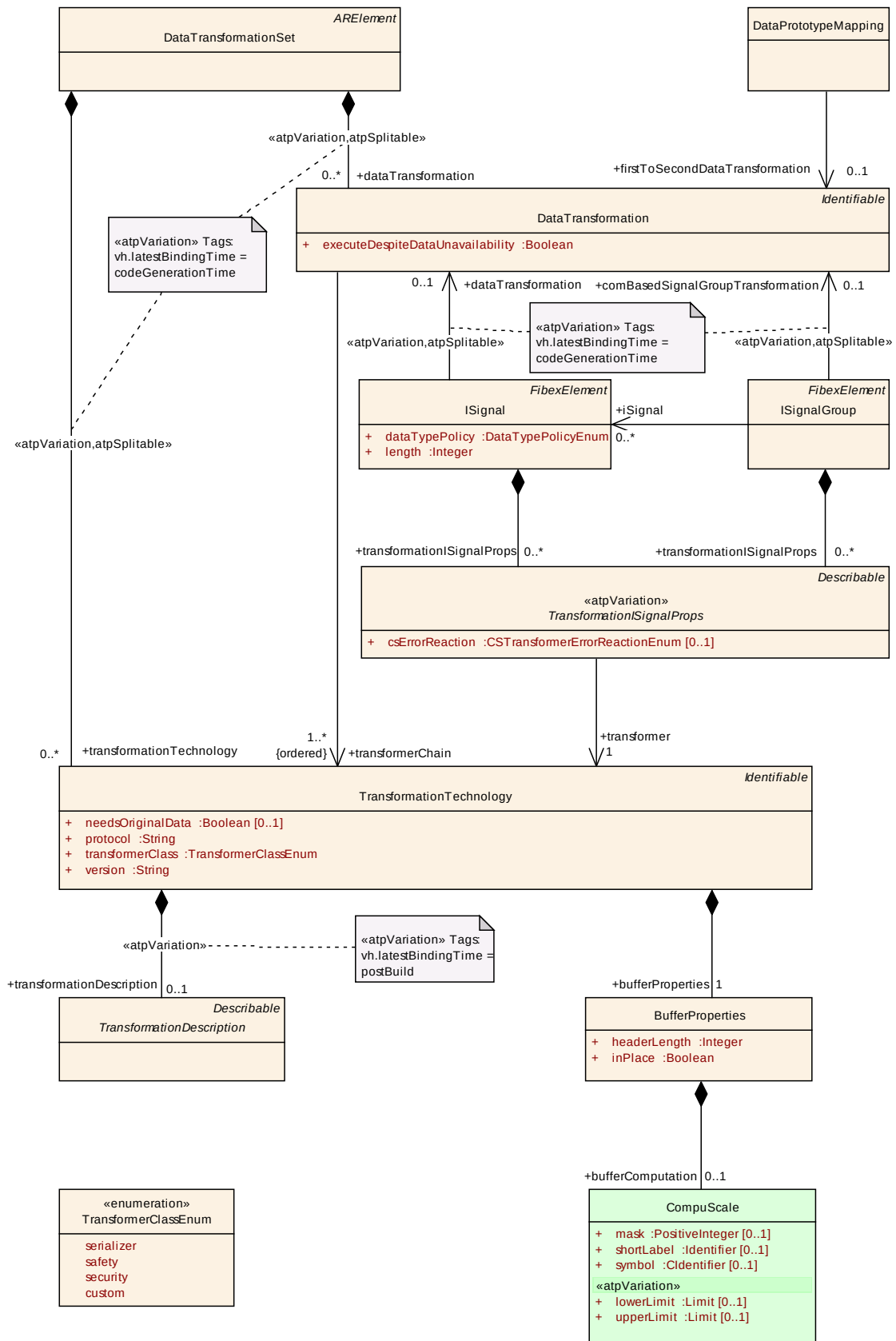
Class	TransformationComSpecProps (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
Note	TransformationComSpecProps holds all the attributes for transformers that are port specific.			
Base	ARObject, Describable			
Attribute	Datatype	Mul.	Kind	Note
—	—	—	—	—

Table 4.82: TransformationComSpecProps

It can be determined by the specific [TransformationComSpecProps](#) to which transformer this configuration is applicable:

- The configuration in [EndToEndTransformationComSpecProps](#) is applicable to E2E transformer (protocol of [TransformationTechnology](#) is set to End-ToEnd).
- The configuration in [UserDefinedTransformationComSpecProps](#) is applicable to a user-defined transformer.

[TPS_SWCT_01598] More than one user-defined transformer is used within one transformer chain [If more than one user-defined transformer is used within one transformer chain (defined by meta-class [TransformationTechnology](#)), the [UserDefinedTransformationComSpecProps](#) shall be assigned to the correct user-defined custom transformer in [TransformationTechnology](#).]([RS_SWCT_03221](#))



[constr_1400] Reference to a specific [DataTransformation](#) [A specific [DataTransformation](#) shall only be referenced by either

- a [DataPrototypeMapping](#) in the role [firstToSecondDataTransformation](#) **or**
- an [ISignal](#) in the role [dataTransformation](#) **or**
- an [ISignalGroup](#) in the role [comBasedSignalGroupTransformation](#)

]()

[constr_1401] Restrictions on the relation between [DataPrototypeMapping](#) and [DataTransformation](#) [A [VariableDataPrototype](#) in the context of a [PortPrototype](#) shall **not** be referenced by a [DataPrototypeMapping](#) that references a [DataTransformation](#) while a [DataMapping](#) exists that points to this [VariableDataPrototype](#) (via the [SystemSignal](#)) that also refers to an [ISignal](#) that in turn references a [DataTransformation](#).]()

In other words: a [VariableDataPrototype](#) can either become a part of a [DataPrototypeMapping](#)-based data transformation or of an [ISignal](#)-based data transformation.

Please note that in a composite software structure the [VariableDataPrototype](#) can be delegated throughout the [CompositionSwComponentType](#) and **[constr_1401]** still applies.

Class	TransformationTechnology			
Package	M2::AUTOSARTemplates::SystemTemplate::Transformer			
Note	A TransformationTechnology is a transformer inside a transformer chain. Tags: xml.namePlural=TRANSFORMATION-TECHNOLOGIES			
Base	ARObject, Identifiable ,MultilanguageReferrable, Referrable			
Attribute	Datatype	Mul.	Kind	Note
bufferProperties	BufferProperties	1	aggr	Aggregation of the mandatory BufferProperties.
needsOriginalData	Boolean	0..1	attr	Specifies whether this transformer gets access to the SWC's original data.
protocol	String	1	attr	Specifies the protocol that is implemented by this transformer.
transformationDescription	TransformationDescription	0..1	aggr	A transformer can be configured with transformer specific parameters which are represented by the TransformerDescription. Stereotypes: atpVariation Tags: vh.latestBindingTime=postBuild
transformerClass	TransformerClassEnum	1	attr	Specifies to which transformer class this transformer belongs.
version	String	1	attr	Version of the implemented protocol.

Table 4.83: TransformationTechnology

Based on the user defined attributes inside [UserDefinedTransformationComSpecProps](#) (which are, of course, not standardized), the generator of the user-defined transformer shall determine to which user-defined transformer a [UserDefinedTransformationComSpecProps](#) belongs to.

Class	UserDefinedTransformationComSpecProps			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
Note	The UserDefinedTransformationComSpecProps is used to specify port specific configuration properties for custom transformers.			
Base	ARObject, Describable , TransformationComSpecProps			
Attribute	Datatype	Mul.	Kind	Note
—	—	—	—	—

Table 4.84: UserDefinedTransformationComSpecProps

[TPS_SWCT_01599] **PortPrototype-specific configuration for custom transformers** [Meta-class [UserDefinedTransformationComSpecProps](#) shall be used for the specification of [PortPrototype](#)-specific configuration options for custom transformers.]([RS_SWCT_03221](#))

Please note that it is possible to add custom configuration items in [UserDefinedTransformationComSpecProps](#) by means of the attribute [adminData.sdq](#).

Class	EndToEndTransformationComSpecProps			
Package	M2::AUTOSARTemplates::SystemTemplate::Transformer			
Note	The class EndToEndTransformationComSpecProps specifies port specific configuration properties for EndToEnd transformer attributes.			
Base	ARObject, Describable , TransformationComSpecProps			
Attribute	Datatype	Mul.	Kind	Note
disableEndToEndCheck	Boolean	1	attr	Disables/Enables the E2E check. The E2Eheader is removed from the payload independent from the setting of this attribute.
maxDeltaCounter	PositiveInteger	0..1	attr	Maximum allowed gap between two counter values of two consecutive checks.
maxErrorStateInit	PositiveInteger	0..1	attr	Maximal number of checks in which ProfileStatus equal to E2E_P_ERROR was determined, within the last WindowSize checks, for the state E2E_SM_INIT. The minimum value is 0.
maxErrorStateInvalid	PositiveInteger	0..1	attr	Maximal number of checks in which ProfileStatus equal to E2E_P_ERROR was determined, within the last WindowSize checks, for the state E2E_SM_INVALID. The minimum value is 0.

Attribute	Datatype	Mul.	Kind	Note
maxErrorStateValid	PositiveInteger	1	attr	Maximal number of checks in which ProfileStatus equal to E2E_P_ERROR was determined, within the last WindowSize checks, for the state E2E_SM_VALID. The minimum value is 0.
maxNewOrRepeatedData	PositiveInteger	0..1	attr	EndToEndTransformationDescription holds these attributes which are profile specific and have the same value for all E2E transformers.
minOkStateInit	PositiveInteger	0..1	attr	Minimal number of checks in which ProfileStatus equal to E2E_P_OK was determined, within the last WindowSize checks, for the state E2E_SM_INIT. The minimum value is 1.
minOkStateInvalid	PositiveInteger	0..1	attr	Minimal number of checks in which ProfileStatus equal to E2E_P_OK was determined, within the last WindowSize checks, for the state E2E_SM_INVALID. The minimum value is 1.
minOkStateValid	PositiveInteger	0..1	attr	Minimal number of checks in which ProfileStatus equal to E2E_P_OK was determined, within the last WindowSize checks, for the state E2E_SM_VALID. The minimum value is 1.
syncCounterInit	PositiveInteger	0..1	attr	EndToEndTransformationDescription holds these attributes which are profile specific and have the same value for all E2E transformers.
windowSize	PositiveInteger	0..1	attr	Size of the monitoring window for the E2E state machine. The meaning is the number of correct cycles (E2E_P_OK) that are required in E2E_SM_INITCOM before the transition to E2E_SM_VALID. The minimum allowed value is 1.

Table 4.85: EndToEndTransformationComSpecProps

[TPS_SWCT_01600] PortPrototype-specific configuration for data transformers related to end-to-end protection [Meta-class [EndToEndTransformationComSpecProps](#) shall be used for the specification of PortPrototype-specific configuration options for data transformers related to end-to-end protection.]([RS_SWCT_03221](#))

4.6 Port Groups within Component Types

[TPS_SWCT_01063] **PortGroup** [A *SwComponentType* can declare that some of its *PortPrototypes* belong to a *PortGroup*.

Such a port group defines a logical grouping of *PortPrototypes* which is used as input to configure the implementation of mode managers in the basic software, for example the communication of bus signals associated with the grouped ports may be suppressed in a certain mode.] (*RS_SWCT_03200*, *RS_SWCT_03201*)

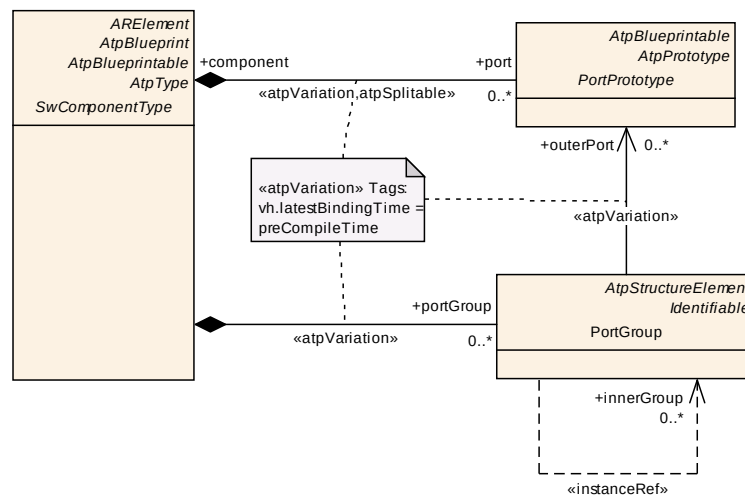


Figure 4.44: Declaration of **PortGroups**

Class	PortGroup			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	Group of ports which share a common functionality, e.g. need specific network resources. This information shall be available on the VFB level in order to delegate it properly via compositions. When propagated into the ECU extract, this information is used as input for the configuration of Services like the Communication Manager. A PortGroup is defined locally in a component (which can be a composition) and refers to the "outer" ports belonging to the group as well as to the "inner" groups which propagate this group into the components which are part of a composition. A PortGroup within an atomic SWC cannot be linked to inner groups.			
Base	ARObject, AtpClassifier, AtpFeature, AtpStructureElement, Identifiable, Multilanguage Referrable, Referrable			
Attribute	Datatype	Mul.	Kind	Note
innerGroup	PortGroup	*	iref	Links a PortGroup in a composition to another PortGroup, that is defined in a component which is part of this CompositionSwComponentType.
outerPort	PortPrototype	*	ref	Outer port of this component which belongs to the group. A port can belong to several groups or to no group at all. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime

Table 4.86: PortGroup

[TPS_SWCT_01064] PortGroups have to be defined on the VFB level [Though the declaration `PortGroups` is not relevant for the RTE, they have to be defined on the VFB level, because they represent design decisions taken on this level. Accordingly, `PortGroups` can be defined for `CompositionSwComponentTypes` as well as for `AtomicSwComponentTypes`.]([RS_SWCT_03200](#), [RS_SWCT_03201](#))

[TPS_SWCT_01065] PortPrototype may belong to more than one PortGroups [A `PortPrototype` may belong to more than one `PortGroups` and `PortGroups` can be associated with the “inner” `PortGroups` of `SwComponentPrototypes` which are aggregated by the same `SwComponentType` as the `PortGroup`. By this, `PortGroups` can be locally defined but still traced down the component hierarchy.]([RS_SWCT_03200](#), [RS_SWCT_03201](#))

[TPS_SWCT_01066] PortGroups can be associated with certain ServiceNeeds [`PortGroups` can be associated with certain `ServiceNeeds` in order to trace the information down to the configuration of the basic software, for details see chapter 7.11.2.]([RS_SWCT_03200](#), [RS_SWCT_03201](#))

[constr_1147] Standardized values for the attribute category of meta-class PortGroup [

The following values of the attribute `category` of meta-class `PortGroup` are reserved by the AUTOSAR standard:

- `MODE_MANAGEMENT`: This represents the usage of the `PortGroup` for the purpose of mode management
- `PARTIAL_NETWORKING`: This represents the usage of the `PortGroup` for the purpose of partial networking

]()

4.7 End to End Protection

The aspect of end-to-end protection has seen different support by the AUTOSAR meta-model.

On the one hand, there is the definition of dedicated meta-classes, e.g. `EndToEndDescription`, which aim at an implementation that uses a so-called E2E wrapper (an approach with a software component above RTE invoking the E2E library) or AUTOSAR Com module callout mechanism (with Com callouts used to invoke E2E library).

This approach is documented in chapter 4.7 of this document.

As an alternative approach, it is possible to implement end-to-end protection using so-called data transformers.

The detailed description of how this approach can be configured is beyond the scope of this document. Please refer to the TPS System Template [11] where the details of the alternative approach are explained.

In contrast to the approach based on the [EndToEndProtection](#) and [EndToEndDescription](#) (which partly involves technologies that are not subjected to the AUTOSAR standard), the second approach is fully standardized by AUTOSAR.

As described in [19] there are cases where safety-related software-components protect the data exchanged between each other. For this purpose modeling support is provided by the software-component template.

Note that several end-to-end profiles are selectable for a specific application. The specific end-to-end profile is represented by the attribute [category](#) of meta-class [EndToEndDescription](#).

Semantically, the [category](#) value represents an identification of the specific end-to-end profile applicable for the communication of the corresponding data element. According to [19] there are two pre-defined profiles that can be used.

[TPS_SWCT_01089] end-to-end communication protection [The information specific to each profile is expressed by the set of attributes of [EndToEndDescription](#) owned by [EndToEndProtection](#) in the role [endToEndProfile](#).] ([RS_SWCT_03240](#))

Class	EndToEndDescription			
Package	M2::AUTOSARTemplates::SWComponentTemplate::EndToEndProtection			
Note	This meta-class contains information about end-to-end protection. The set of applicable attributes depends on the actual value of the category attribute of EndToEndProtection.			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
category	NameToken	1	attr	The category represents the identification of the concrete E2E profile. The applicable values are specified in a semantic constraint and determine the applicable attributes of EndToEndDescription. Tags: xml.sequenceOffset=-100
counterOffset	PositiveInteger	0..1	attr	Bit offset of Counter from the beginning of the Array representation of the Signal Group/VariableDataPrototype (MSB order, bit numbering: bit 0 is the least important). The offset shall be a multiplicity of 4 and it should be 8 whenever possible. For example, offset 8 means that the counter will take the low nibble of the byte 1, i.e. bits 8 .. 11. If counterOffset is not present the value is defined by the selected profile. Tags: xml.sequenceOffset=-50

Attribute	Datatype	Mul.	Kind	Note
crcOffset	PositiveInteger	0..1	attr	Bit offset of CRC from the beginning of the Array representation of the Signal Group/VariableDataPrototype (MSB order, bit numbering: bit 0 is the least important). The offset shall be a multiplicity of 8 and it should be 0 whenever possible. For example, offset 8 means that the CRC will take the byte 1, i.e. bits 8..15. If crcOffset is not present the value is defined by the selected profile. Tags: xml.sequenceOffset=-60
dataId (ordered)	PositiveInteger	*	attr	This represents a unique numerical identifier. Note: ID is used for protection against masquerading. The details concerning the maximum number of values (this information is specific for each E2E profile) applicable for this attribute are controlled by a semantic constraint that depends on the category of the EndToEndProtection. Tags: xml.sequenceOffset=-90
dataIdMode	PositiveInteger	0..1	attr	There are three inclusion modes how the implicit two-byte Data ID is included in the one-byte CRC: • dataIdMode = 0: Two bytes are included in the CRC (double ID configuration) This is used in variant 1A. • dataIdMode = 1: One of the two bytes byte is included, alternating high and low byte, depending on parity of the counter (alternating ID configuration). For even counter low byte is included; For odd counters the high byte is included. This is used in variant 1B. • dataIdMode = 2: Only low byte is included, high byte is never used. This is applicable if the IDs in a particular system are 8 bits. • dataIdMode = 3: The low byte is included in the implicit CRC calculation, the low nibble of the high byte is transmitted along with the data (i.e. it is explicitly included), the high nibble of the high byte is not used. This is applicable for the IDs up to 12 bits. Tags: xml.sequenceOffset=-85
dataIdNibbleOffset	PositiveInteger	0..1	attr	Bit offset of the low nibble of the high byte of Data ID. The applicability of this attribute is controlled by [constr_1261]. Tags: xml.sequenceOffset=-25

Attribute	Datatype	Mul.	Kind	Note
dataLength	PositiveInteger	0..1	attr	This attribute represents the length of the Array representation of the Signal Group/VariableDataPrototype including CRC and Counter in bits. Tags: xml.sequenceOffset=-80
maxDeltaCounterInit	PositiveInteger	0..1	attr	Initial maximum allowed gap between two counter values of two consecutively received valid Data, i.e. how many subsequent lost data is accepted. For example, if the receiver gets Data with counter 1 and MaxDeltaCounterInit is 1, then at the next reception the receiver can accept Counters with values 2 and 3, but not 4. Note that if the receiver does not receive new Data at a consecutive read, then the receiver increments the tolerance by 1. Tags: xml.sequenceOffset=-70
maxNumberOfRepeatedData	PositiveInteger	0..1	attr	The maximum amount of missing or repeated Data which the receiver does not expect to exceed under normal communication conditions. Tags: xml.sequenceOffset=-40
syncCounterInit	PositiveInteger	0..1	attr	Number of Data required for validating the consistency of the counter that shall be received with a valid counter (i.e. counter within the allowed lock-in range) after the detection of an unexpected behavior of a received counter. Tags: xml.sequenceOffset=-30

Table 4.87: EndToEndDescription

[TPS_SWCT_01090] **EndToEndProtection** [[EndToEndProtection](#) is the [Identifiable](#) class that owns specific elements for referencing the to-be-protected data elements and signals

- [EndToEndProtectionVariablePrototype](#): a specific [dataElement](#) owned by a specific [PortPrototype](#)
- [EndToEndProtectionISignalIPdu](#): a specific [ISignalGroup](#) in the context of an [ISignalIPdu](#). For more details please refer to [11]

]([RS_SWCT_03240](#))

[TPS_SWCT_01091] **Two cases for end-to-end protection** [In order to protect a [VariableDataPrototype](#) the [EndToEndProtectionVariablePrototype](#) shall be defined. If communication is defined between ECUs using AUTOSAR COM the [EndToEndProtectionISignalIPdu](#) shall be defined as well.]([RS_SWCT_03240](#))

The following features apply:

- **[constr_1000] End-to-end protection is limited to sender/receive communication** [end-to-end protection applies for sender/receiver communication only]()
- The value of the `dataId` is assigned by a central authority rather than by the developer of the software-component.
- The information about the `dataId` shall be available at both the sender and the receiver(s).
- **[constr_1001] Value of `dataId` shall be unique** [The value of the `dataId` shall be unique within the scope of the `System`.]()
- **[TPS_SWCT_01508] Scope of end-to-end protection** [End-to-end protection applies to local (i.e. within the ECU) as well as remote (i.e. ECU to ECU) communication.]([RS_SWCT_03240](#))

[TPS_SWCT_01092] `EndToEndProtectionSet` [The meta-class `EndToEndProtectionSet` provides a container for `EndToEndProtection`. The aggregation is stereotyped `<<atpSplitable>>` because the information about end-to-end protection is added at a later step in the development workflow.]([RS_SWCT_03240](#))

It also has the stereotype `<<atpVariation>>` because this allows for implementing the software-component in two variants, one that uses end-to-end protection and one that does not use it. It also might happen that the communication ends themselves are variant.

`EndToEndProtection` maintains `InstanceRefs` to one `dataElement` in the role of `sender` and to one or many `dataElements` in the role of `receiver`. By this means it is possible to support a 1:n communication scenario.

[TPS_SWCT_01093] Definition of end-to-end protection is splitable [`EndToEndProtection` aggregates `EndToEndDescription` using stereotype `<<atpSplitable>>`. By this means it is for the integrator of an ECU possible to generally specify the nature of a specific end-to-end protection but leave the actual assignment of values (e.g. for `dataId`) to a later process step.]([RS_SWCT_03240](#))

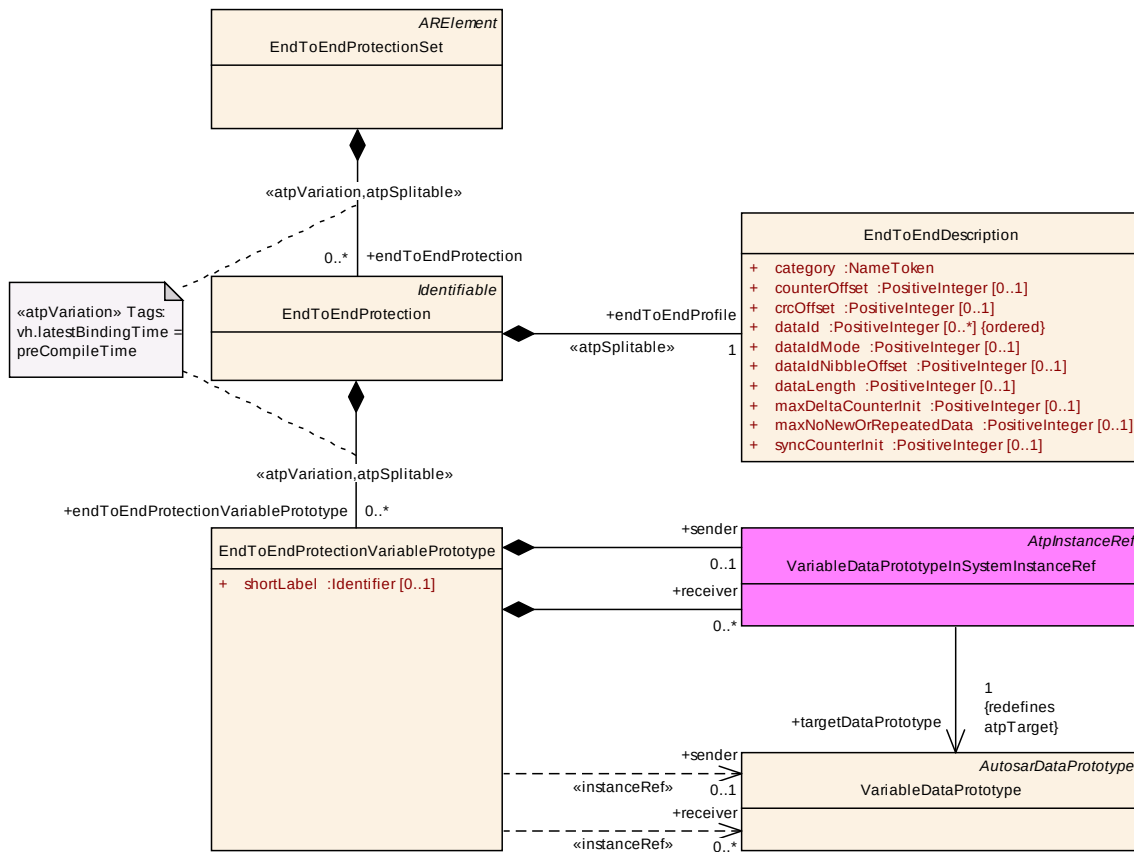


Figure 4.45: Details of the modeling of end-to-end protection

According to [19] the following constraints apply on the attributes of `EndToEndProtection` (note that additional M1 constraints apply as described in [19]):

[constr_1110] Value of `category` in `EndToEndDescription` [The attribute `category` of `EndToEndDescription` can have the following values:

- NONE
- PROFILE_01
- PROFILE_02

]()

[TPS_SWCT_01094] `category` of `EndToEndDescription` [The values for the `category` of `EndToEndDescription` mentioned in [constr_1110] are standardized and reserved for being used in the way the AUTOSAR standard foresees. In addition, it is positively possible to use other than the standardized values for the `category`.](*RS_SWCT_03240*)

This aspect will be clarified in more detail in later revisions of the AUTOSAR standard. For the time being, it shall be noted that the usage of other than the standardized values shall not create name clashes with future standardized values. This can be achieved by using e.g. a company-specific prefix or suffix to the value of `category`.

The semantics of the `category`s is:

NONE this indicates that the E2E framework shall be enabled for the given `sender/receiver` respectively the given `iSignalIPdu`. The wrapper code shall be generated but it shall not invoke E2E library protection routines. E2E wrapper works as pass-through.

This may be used when a profile selection or profile options are not yet selected in a given system but it is required that the system can be built successfully under consideration of the E2E library. This would also be applicable for migrating from/to a system with/without E2E protection.

[TPS_SWCT_01095] category set to NONE [If attributes exist in the presence of the `category` being set to NONE the attributes shall be ignored.]([RS_SWCT_03240](#))

PROFILE_01 This indicates that the settings of E2E profile 1 (that uses a SAE CRC8, implicit 16 bit data ID, and a 4 bit alive counter) apply.

[constr_1113] Existence of attributes in PROFILE_01 [In PROFILE_01, the following attributes shall exist:

- `dataLength`
- `dataId`

]()

Please note that the attribute `maxDeltaCounterInit` is also part of PROFILE_01 but it does not necessarily have to exist provided that `ReceiverComSpec.maxDeltaCounterInit` exists.

[constr_1170] Interpretation of attribute maxDeltaCounterInit owned by EndToEndDescription [If `EndToEndProtection.endToEndProtectionVariablePrototype.receiver` is identical to the `RPortPrototype.requiredComSpec.dataElement` and `RPortPrototype.requiredComSpec.maxDeltaCounterInit` is defined then the value of `RPortPrototype.requiredComSpec.maxDeltaCounterInit` shall be preferred over the value of `EndToEndProtection.endToEndProfile.maxDeltaCounterInit`.

If the value of `category` of `EndToEndDescription` is set to PROFILE_01 and either the described correspondence rule concerning the referenced `VariableDataPrototype` is not fulfilled or `RPortPrototype.requiredComSpec.maxDeltaCounterInit` is not defined then `EndToEndProtection.endToEndProfile.maxDeltaCounterInit` shall exist.]()

[constr_1111] Constraints of dataId in PROFILE_01 [In PROFILE_01, there shall be only one element in the set and the applicable range of values is [0 .. 65535].]()

[constr_1112] Constraints of `dataIdMode` in PROFILE_01 [In PROFILE_01, the applicable range of values for `dataIdMode` is [0 .. 3].]()

[constr_1114] Constraints of `crcOffset` in PROFILE_01 [In PROFILE_01, the applicable range of values for `crcOffset` is [0 .. 65535]. For the value of this attribute the constraint *value mod 4 = 0* applies.]()

[constr_1115] Constraints of `counterOffset` in PROFILE_01 [In PROFILE_01, the applicable range of values for `counterOffset` is [0 .. 65535]. For the value of this attribute the constraint *value mod 4 = 0* applies.]()

[constr_1116] Constraints of `dataLength` in PROFILE_01 [In PROFILE_01, the applicable range of values for `dataLength` is [0 .. 240]. For the value of this attribute the constraint *value mod 8 = 0* applies.]()

[constr_1117] Constraints of `maxDeltaCounterInit` in PROFILE_01 [In PROFILE_01, the applicable range of values for `EndToEndDescription.maxDeltaCounterInit` and `ReceiverComSpec.maxDeltaCounterInit` is [0 .. 14].]()

[constr_1211] Constraints of `maxNoNewOrRepeatedData` in PROFILE_01 [In PROFILE_01, the applicable range of values for `EndToEndDescription.maxNoNewOrRepeatedData` and `ReceiverComSpec.maxNoNewOrRepeatedData` is [0 .. 14].]()

[constr_1212] Constraints of `syncCounterInit` in PROFILE_01 [In PROFILE_01, the applicable range of values for `EndToEndDescription.syncCounterInit` and `ReceiverComSpec.syncCounterInit` is [0 .. 14].]()

[constr_1215] Interpretation of attribute `maxNoNewOrRepeatedData` owned by `EndToEndDescription` in PROFILE_01 [If `EndToEndProtection.endToEndProtectionVariablePrototype.receiver` is identical to the `RPortPrototype.requiredComSpec.dataElement` and `RPortPrototype.requiredComSpec.maxNoNewOrRepeatedData` is defined then the value of `RPortPrototype.requiredComSpec.maxNoNewOrRepeatedData` shall be preferred over the value of `EndToEndProtection.endToEndProfile.maxNoNewOrRepeatedData`.]()

If the value of `category` of `EndToEndDescription` is set to PROFILE_01 and either the described correspondence rule concerning the referenced `VariableDataPrototype` is not fulfilled or `RPortPrototype.requiredComSpec.maxNoNewOrRepeatedData` is not defined then `EndToEndProtection.endToEndProfile.maxNoNewOrRepeatedData` shall exist.]()

[constr_1216] Interpretation of attribute `syncCounterInit` owned by `EndToEndDescription` in PROFILE_01 [If `EndToEndProtection.endToEndProtectionVariablePrototype.receiver` is identical to the `RPortPrototype.requiredComSpec.dataElement` and `RPortPrototype.requiredComSpec.syncCounterInit` is defined then the value of `RPortPro-`

`prototype.requiredComSpec.syncCounterInit` **shall be preferred** over the value of `EndToEndProtection.endToEndProfile.syncCounterInit`.

If the value of `category` of `EndToEndDescription` is set to `PROFILE_01` **and either** the described correspondence rule concerning the referenced `VariableDataPrototype` is not fulfilled **or** `RPortPrototype.requiredComSpec.syncCounterInit` is not defined **then** `EndToEndProtection.endToEndProfile.syncCounterInit` **shall exist**. `]()`

[constr_1261] Applicability for `EndToEndDescription.dataIdNibbleOffset` `[` `EndToEndDescription.dataIdNibbleOffset` shall be used **only** if `EndToEndDescription.dataIdMode` is set to the value 3 **and** at the same time `EndToEndDescription.category` is set to `PROFILE_01`. `]`

[TPS_SWCT_01529] Default value for `EndToEndDescription.dataIdNibbleOffset` `[` If `EndToEndDescription.dataIdMode` is set to the value 3 **and** at the same time `EndToEndDescription.category` is set to the value `PROFILE_01` **and** `EndToEndDescription.dataIdNibbleOffset` is not specified, then the default value of 12 (bits) shall be assumed for the attribute `EndToEndDescription.dataIdNibbleOffset`. `]` *(RS_SWCT_03240)*

PROFILE_02 this indicates that the settings of E2E profile 2 apply.

[constr_1118] Existence of attributes in PROFILE_02 `[` In `PROFILE_02`, only the following attributes shall exist:

- `dataLength`
- `dataId`

`]`

Please note that the attribute `maxDeltaCounterInit` is also part of `PROFILE_01` but it does not necessarily have to exist provided that `ReceiverComSpec.maxDeltaCounterInit` exists.

[constr_1171] Interpretation of attribute `maxDeltaCounterInit` of `EndToEndDescription` `[` If `EndToEndProtection.endToEndProtectionVariablePrototype.receiver` is identical to the `RPortPrototype.requiredComSpec.dataElement` **and** `RPortPrototype.requiredComSpec.maxDeltaCounterInit` is defined **then** the value of `RPortPrototype.requiredComSpec.maxDeltaCounterInit` **shall be preferred** over the value of `EndToEndProtection.endToEndProfile.maxDeltaCounterInit`.

If the value of `category` of `EndToEndDescription` is set to `PROFILE_02` **and either** the described correspondence rule concerning the referenced `VariableDataPrototype` is not fulfilled **or** `RPortPrototype.requiredComSpec.maxDeltaCounterInit` is not defined **then** `EndToEndProtection.endToEndProfile.maxDeltaCounterInit` **shall exist**. `]()`

[constr_1119] Constraints of `dataLength` in `PROFILE_02` [In `PROFILE_02`, the applicable range of values for `dataLength` is [0 .. 65535]. For the value of this attribute the constraint *value mod 8 = 0* applies.]()

[constr_1120] Constraints of `dataId` in `PROFILE_02` [In `PROFILE_02`, there shall be exactly ordered 16 elements in the set and the applicable range of values is [0 .. 255].]()

[constr_1121] Constraints of `maxDeltaCounterInit` in `PROFILE_02` [In `PROFILE_02`, the applicable range of values for `EndToEndDescription.maxDeltaCounterInit` and `ReceiverComSpec.maxDeltaCounterInit` is [0 .. 15].]()

[constr_1213] Constraints of `maxNoNewOrRepeatedData` in `PROFILE_02` [In `PROFILE_02`, the applicable range of values for `EndToEndDescription.maxNoNewOrRepeatedData` and `ReceiverComSpec.maxNoNewOrRepeatedData` is [0 .. 15].]()

[constr_1214] Constraints of `syncCounterInit` in `PROFILE_02` [In `PROFILE_02`, the applicable range of values for `EndToEndDescription.syncCounterInit` and `ReceiverComSpec.syncCounterInit` is [0 .. 15].]()

[constr_1217] Interpretation of attribute `maxNoNewOrRepeatedData` owned by `EndToEndDescription` in `PROFILE_02` [If `EndToEndProtection.endToEndProtectionVariablePrototype.receiver` is identical to the `RPortPrototype.requiredComSpec.dataElement` and `RPortPrototype.requiredComSpec.maxNoNewOrRepeatedData` is defined then the value of `RPortPrototype.requiredComSpec.maxNoNewOrRepeatedData` shall be preferred over the value of `EndToEndProtection.endToEndProfile.maxNoNewOrRepeatedData`.

If the value of `category` of `EndToEndDescription` is set to `PROFILE_02` and either the described correspondence rule concerning the referenced `VariableDataPrototype` is not fulfilled or `RPortPrototype.requiredComSpec.maxNoNewOrRepeatedData` is not defined then `EndToEndProtection.endToEndProfile.maxNoNewOrRepeatedData` shall exist.]()

[constr_1218] Interpretation of attribute `syncCounterInit` owned by `EndToEndDescription` in `PROFILE_02` [If `EndToEndProtection.endToEndProtectionVariablePrototype.receiver` is identical to the `RPortPrototype.requiredComSpec.dataElement` and `RPortPrototype.requiredComSpec.syncCounterInit` is defined then the value of `RPortPrototype.requiredComSpec.syncCounterInit` shall be preferred over the value of `EndToEndProtection.endToEndProfile.syncCounterInit`.

If the value of `category` of `EndToEndDescription` is set to `PROFILE_02` and either the described correspondence rule concerning the referenced `VariableDataPrototype` is not fulfilled or `RPortPrototype.requiredComSpec.syncCounterInit` is not defined then `EndToEndProtection.endToEndProfile.syncCounterInit` shall exist.]()

Class	EndToEndProtectionSet			
Package	M2::AUTOSARTemplates::SWComponentTemplate::EndToEndProtection			
Note	This represents a container for collection EndToEndProtectionInformation. Tags: atp.recommendedPackage=EndToEndProtectionSets			
Base	ARElement,ARObject,CollectableElement,Identifiable,MultilanguageReferrable,PackageableElement,Referrable			
Attribute	Datatype	Mul.	Kind	Note
endToEndProtection	EndToEndProtection	*	aggr	This is one particular EndToEndProtection. Stereotypes: atpSplitable; atpVariation Tags: atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime

Table 4.88: EndToEndProtectionSet

Class	EndToEndProtection			
Package	M2::AUTOSARTemplates::SWComponentTemplate::EndToEndProtection			
Note	This meta-class represents the ability to describe a particular end to end protection.			
Base	ARObject,Identifiable,MultilanguageReferrable,Referrable			
Attribute	Datatype	Mul.	Kind	Note
endToEndProfile	EndToEndDescription	1	aggr	This represents the particular EndToEndDescription. Stereotypes: atpSplitable Tags: atp.Splitkey=endToEndProfile
endToEndProtectionISignalPdu	EndToEndProtectionISignalPdu	*	aggr	Defines to which ISignalPdu - ISignalGroup pair this EndToEndProtection shall apply. In case several ISignalGroups are used to transport the data (e.g. fan-out in the RTE) there may exist several EndToEndProtectionISignalPdu definitions. Stereotypes: atpSplitable; atpVariation Tags: atp.Splitkey=variationPoint.shortLabel vh.latestBindingTime=preCompileTime

Attribute	Datatype	Mul.	Kind	Note
endToEndProtectionVariablePrototype	EndToEndProtectionVariablePrototype	*	aggr	Defines to which VariableDataPrototypes in the roles of one sender and one or more receivers this EndToEndProtection applies. It shall be possible to aggregate several EndToEndProtectionVariablePrototype in case additional hierarchical decompositions are introduced subsequently. In this case one particular PortPrototype is split into multiple PortPrototypes and connectors, all representing the same data entity. Stereotypes: atpSplittable; atpVariation Tags: atp.Splitkey=shortLabel, variationPoint.shortLabel vh.latestBindingTime=preCompileTime

Table 4.89: EndToEndProtection

Class	EndToEndProtectionVariablePrototype			
Package	M2::AUTOSARTemplates::SWComponentTemplate::EndToEndProtection			
Note	It is possible to protect the data exchanged between software components. For this purpose, for each communication to be protected, the user defines a separate EndToEndProtection (specifying a set of protection settings) and refers to a variableDataPrototype in the role of sender and to one or many variableDataPrototypes in the role of receiver. For details, see EndToEnd Library.			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
receiver	VariableDataPrototype	*	iref	This represents the receiver. Note that 1:n communication is supported for this use case.
sender	VariableDataPrototype	0..1	iref	This represents the sender. Can be optional if an ecu extract is provided and the sender is part of the extract.
shortLabel	Identifier	0..1	ref	This serves as part of the split key in case of more than one EndToEndProtectionVariablePrototype is aggregated in the bound model.

Table 4.90: EndToEndProtectionVariablePrototype

Please note that using end-to-end protection it is explicitly supported that one sender may correspond to one or more receivers.

[constr_1183] EndToEndProtectionVariablePrototypes aggregated by EndToEndProtection | All EndToEndProtectionVariablePrototypes aggregated by the same EndToEndProtection shall refer to the identical sender. |()

4.8 Partial Networking

[TPS_SWCT_01169] Support for partial networking [On the level of the Software Component Template, partial networking is supported by means of the concept of a “Virtual Function Cluster” (VFC). The latter groups all communication on the VFB with respect to a given function. However, the conceptual idea of a Virtual Function Cluster is not represented in the meta-model as such. Instead, [PortGroups](#) (see chapter 4.6) are used to specify the grouping of [PortPrototypes](#) to the higher conceptual level of a Virtual Function Cluster.]([RS_SWCT_03241](#), [RS_SWCT_03201](#))

There are no restrictions regarding the structure of [PortGroup](#) definitions on M1. One [PortPrototype](#) may become a member of several [PortGroups](#), thereby creating overlapping [PortGroups](#).

[TPS_SWCT_01170] Purpose of Virtual Function Cluster [The purpose of Virtual Function Cluster within the Software Component Template mainly has three aspects:

1. assign [PortPrototypes](#) (non service related) of Sender Receiver or Client Server communication to Virtual Function Clusters.
2. control the behavior of the corresponding function in terms of whether or not it is required at a given point in time. This aspect is implemented by the concept of a **control port**. Software-components that implement control ports of a Virtual Function Cluster conceptually become **VFC Controllers**.
3. allow for the application software to retrieve the status of a given Virtual Function Cluster. This aspect is implemented by the concept of a **status port**.

]([RS_SWCT_03241](#))

The usage of the generic concept of [PortGroups](#) for the purpose of partial networks shall be indicated by setting the value of the attribute [category](#) of [PortGroup](#) to `PARTIAL_NETWORKING`.

4.8.1 VFC Control Ports

[TPS_SWCT_01171] Purpose of a control port [The purpose of a control port is to request or release a VFC. Requesting means that the VFC is actively using communication resources while *release* boils down to the VFC being inactive, i.e. the corresponding partial network may be shut down until further notice.

As the requesting and releasing semantics is implemented by means of interfacing the BSW the corresponding control ports need to be typed by a [PortInterface](#) that has the attribute [isService](#) set to `true`.]([RS_SWCT_03241](#))

[TPS_SWCT_01172] Requesting and releasing partial networks [For requesting and releasing partial networks, the BSW can be interfaced in two alternative (i.e. either one or the other) ways:

- **ComM:** [ClientServerInterface](#) using the standardized `ComM_UserRequest.RequestComMode` [20]
- **BswM:** [SenderReceiverInterface](#) using the standardized `AppModeRequestInterface.requestedMode` [15]

]([RS_SWCT_03241](#))

[TPS_SWCT_01173] Control port shall not become a part of the [PortGroup](#) [Please note that the control port shall **not** become a part of the [PortGroup](#) that defines the particular VFC the control port is going to service.

The relationship is implemented by means of a specific [SwcServiceDependency](#) that owns a [RoleBasedPortAssignment](#) to the intended control port.]([RS_SWCT_03241](#), [RS_SWCT_03201](#))

4.8.2 VFC Status Ports

[TPS_SWCT_01175] Actively query the status of a partial network [Very much like mode management, the concept of partial networking supports the ability to actively query the status of a partial network.

This can be done by means of interfacing the BSW in three alternative (as in “one of”) ways:

- **ComM:** [ClientServerInterface](#) using the standardized `ComM_UserRequest.GetCurrentComMode` [20]
- **ComM:** [ModeSwitchInterface](#) using the standardized `ComM_CurrentMode.currentMode` [20]
- **BswM:** [ModeSwitchInterface](#) using the standardized `AppModeInterface.currentMode` [15]

]([RS_SWCT_03241](#))

As mentioned above, the status of the ComM can be retrieved by either a [ClientServerInterface](#) or a [SenderReceiverInterface](#). Which of the two alternatives applies in a specific case is up to the author of a software-component⁷.

When using one of the possible [SenderReceiverInterfaces](#), the correspondence of the status port concept with mode management extends to the point that the status of the partial network is returned as an actual [ModeDeclaration](#).

This implies that all mechanisms foreseen by the Software Component Template to react on mode changes are in place and can be used within the application software.

⁷The usage of the [ClientServerInterface](#) effectively implements a “pull” approach for the mode information while the usage of the [SenderReceiverInterface](#) resembles a “push” approach if it is used in combination with a [SwcModeSwitchEvent](#).

To assure that the communication via `PortPrototypes` that belong to a partial network is valid the software component shall consider the status of the partial network before communicating in order to assert its activity.

[TPS_SWCT_01174] Status port shall not become a member of the `PortGroup` [A status port shall **not** become a member of the `PortGroup` that corresponds to the partial network subject to the status port.

The relationship is implemented by means of a specific `SwcServiceDependency` that owns a `RoleBasedPortAssignment` to the intended status port.
]([RS_SWCT_03241](#), [RS_SWCT_03201](#))

4.9 Formal Definition of implicit Communication Behavior

[TPS_SWCT_01509] Implicit communication behavior [The purpose of the formal definition of the behavior of a `SwComponentType` with respect to the *implicit* communication can conceptually condensed to two basic aspects:

- **Stable** data during the execution of a group of `RunnableEntity`s. This means that all data values read by different `RunnableEntity`s are from the same age. Therefore the value is not changing during the execution of the chain of `RunnableEntity`s.
- **Coherent** data consumption and propagation for a group of `DataPrototypes`. This means that a set of interdependent data values are from the same calculation iteration. Therefore the set of values has to be propagated at once to `RunnableEntity`s requiring the complete result of the calculation. `RunnableEntity`s which are part of the calculation chain may still consume partly updated values.

]([RS_SWCT_03065](#))

[TPS_SWCT_01481] The meaning of the term *stability* with respect to **ConsistencyNeeds** [The meaning of the term *stability* is that the values of a group of `VariableDataPrototypes` shall not change values during the execution of a group of `RunnableEntity`s.]([RS_SWCT_03065](#))

[TPS_SWCT_01482] The meaning of the term *coherence* with respect to **ConsistencyNeeds** [The meaning of the term *coherence* means that the values of a group of `VariableDataPrototypes` shall not be read by receiving `RunnableEntity`s until all the producing `RunnableEntity`s are terminated.]([RS_SWCT_03065](#))

In response to these goals the meta-model provides means to express the correlation between a group of `RunnableEntity`s and a group of `DataPrototypes`. These groups might be defined **hierarchically**.

The information (in terms of `ConsistencyNeeds`) can be defined primarily during the design of an `AtomicSwComponentType` but it is just as well possible to specify this `ConsistencyNeeds` during the definition of `CompositionSwComponentTypes`.

For example, the existence of stable data is typically expected for the execution of `RunnableEntity`s of several `AtomicSwComponentTypes`.

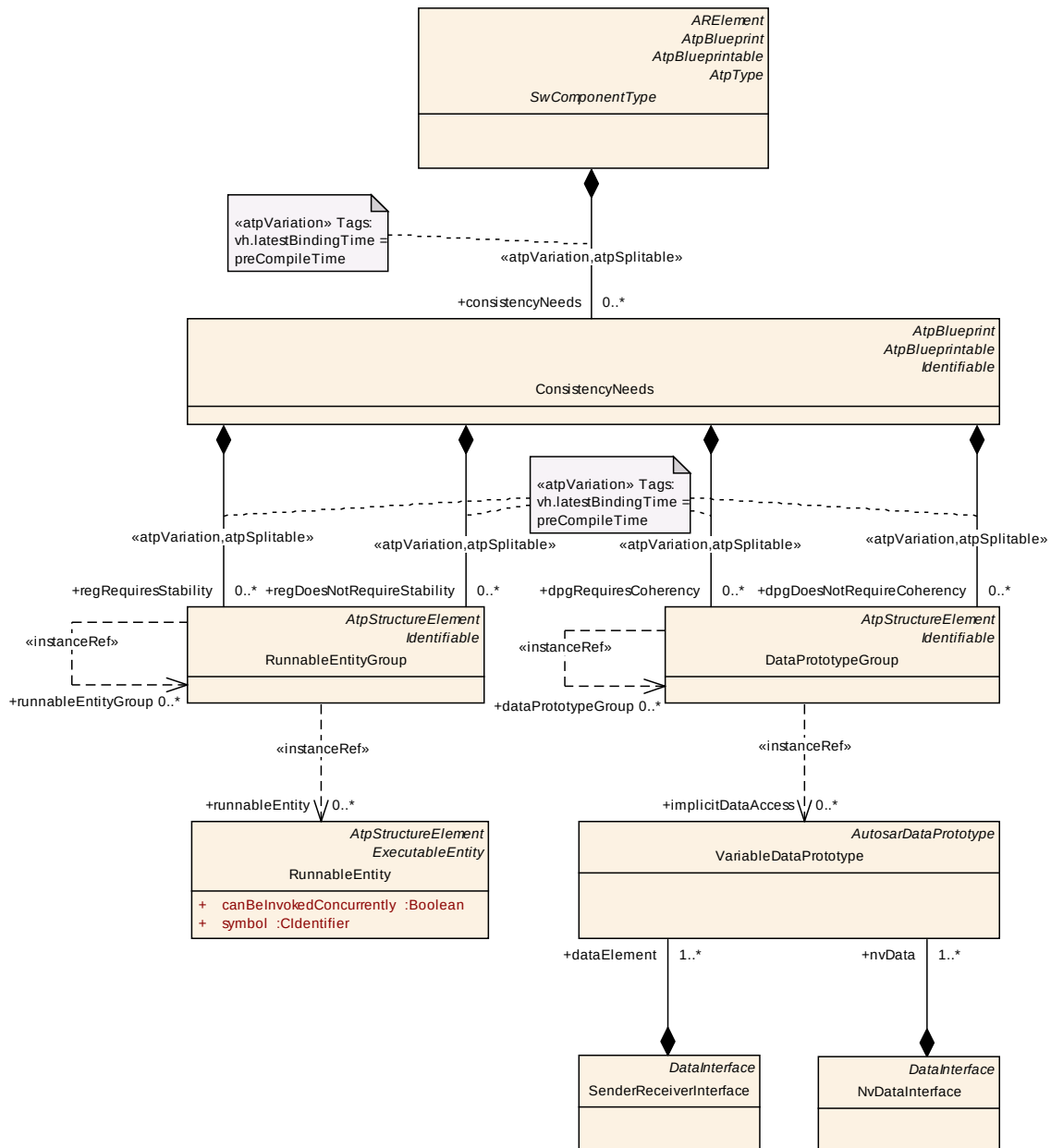


Figure 4.46: Formal definition of implicit communication behavior

Please note that the two aspects *stability* and *coherence* are not necessarily connected to each other. It is possible to require *stability* without *coherence* and vice versa. For this purpose the roles `dpgDoesNotRequireCoherence` and `regDoesNotRequireStability` are needed.

[TPS_SWCT_01480] Stability and/or coherence is not required [In order to be able to clearly separate the aspect of *stability* from *coherence* it is possible to use the roles `dpgDoesNotRequireCoherence` to express that a group of `VariableDataPrototypes` explicitly does not require *consistency*.

Likewise, `regDoesNotRequireStability` can be used to express that for a group of `RunnableEntity`s *stability* with respect to data access is not required. `]()`

[TPS_SWCT_01479] Applicability of `ConsistencyNeeds` `[ ConsistencyNeeds` can only be applied to `RunnableEntity`s that make use of “implicit” communication. `](RS_SWCT_03065)`

[TPS_SWCT_01466] `ConsistencyNeeds` applied on `RunnableEntity`s that do not use implicit communication `[ If a ConsistencyNeeds is applied on RunnableEntitys that do not use implicit communication it shall be ignored. ](RS_SWCT_03065)`

The formal definition of the implicit communication behavior foresees the grouping of model elements in order to indicate their relevance for consistent implicit communication.

[TPS_SWCT_01470] `RunnableEntityGroup` `[A RunnableEntitys belongs to a specific RunnableEntityGroup if it is associated either directly with the given RunnableEntityGroup or if the RunnableEntityGroup the RunnableEntity belongs to is eventually (there can be more than one nesting level) referenced by the given RunnableEntityGroup.](RS_SWCT_03065)`

[TPS_SWCT_01471] `DataPrototypeGroup` `[A VariableDataPrototypes belongs to a specific DataPrototypeGroup if it is associated either directly with the given DataPrototypeGroup or if the DataPrototypeGroup the VariableDataPrototype belongs to is eventually (there can be more than one nesting level) referenced by the given DataPrototypeGroup.](RS_SWCT_03065)`

[constr_1231] `ConsistencyNeeds` aggregated by `CompositionSwComponentType` `[ If ConsistencyNeeds are aggregated by a CompositionSwComponentType the associations stereotyped <<instanceRef>> may only refer to context and target elements within the context of this CompositionSwComponentType. ]()`

For clarification, `[constr_1231]` includes `VariableDataPrototypes` owned by delegation `PortPrototypes` of the owning `CompositionSwComponentType`, `VariableDataPrototypes` in delegation `PortPrototypes` of `CompositionSwComponentType` instantiated in the enclosing `CompositionSwComponentType`, or `VariableDataPrototypes` in `PortPrototypes` owned by `AtomicSwComponentTypes` instantiated inside the context of the enclosing `CompositionSwComponentType`.

[constr_1232] `ConsistencyNeeds` aggregated by `AtomicSwComponentType` `[ If ConsistencyNeeds are aggregated by a AtomicSwComponentType the associations stereotyped <<instanceRef>> may only refer to context and target elements within the context of this AtomicSwComponentType. ]()`

Strictly speaking, these are the `RunnableEntity`s and `PortPrototypes` of this particular `AtomicSwComponentType` or `RunnableEntityGroups` and `DataPrototypeGroups` which are owned by the same `AtomicSwComponentType`.

Please note that pre-defined values for the `category` of `RunnableEntityGroup` and `DataPrototypeGroup` are described in [1].

Class	ConsistencyNeeds			
Package	M2::AUTOSARTemplates::SWComponentTemplate::ImplicitCommunicationBehavior			
Note	This meta-class represents the ability to define requirements on the implicit communication behavior.			
Base	ARObject, AtpBlueprint , AtpBlueprintable , Identifiable , Multilanguage Referrable , Referrable			
Attribute	Datatype	Mul.	Kind	Note
dpgDoesNotRequireCoherency	DataPrototypeGroup	*	aggr	This group of VariableDataPrototypes does not require coherency with respect to the implicit communication behavior. Stereotypes: atpSplitable; atpVariation Tags: atp.Splitkey=shortName, variation Point.shortLabel vh.latestBindingTime=preCompileTime
dpgRequiresCoherency	DataPrototypeGroup	*	aggr	This group of VariableDataPrototypes requires coherency with respect to the implicit communication behavior, i.e. all read and write access to VariableDataPrototypes in the DataPrototypeGroup by the RunnableEntitys of the RunnableEntityGroup need to be handled in a coherent manner. Stereotypes: atpSplitable; atpVariation Tags: atp.Splitkey=shortName, variation Point.shortLabel vh.latestBindingTime=preCompileTime
regDoesNotRequireStability	RunnableEntityGroup	*	aggr	This group of RunnableEntities does not require stability with respect to the implicit communication behavior. Stereotypes: atpSplitable; atpVariation Tags: atp.Splitkey=shortName, variation Point.shortLabel vh.latestBindingTime=preCompileTime
regRequiresStability	RunnableEntityGroup	*	aggr	This group of RunnableEntities requires stability with respect to the implicit communication behavior, i.e. all read and write access to VariableDataPrototypes in the DataPrototypeGroup by the RunnableEntitys of the RunnableEntityGroup need to be handled in a stable manner. Stereotypes: atpSplitable; atpVariation Tags: atp.Splitkey=shortName, variation Point.shortLabel vh.latestBindingTime=preCompileTime

Table 4.91: ConsistencyNeeds

Class	RunnableEntityGroup			
Package	M2::AUTOSARTemplates::SWComponentTemplate::ImplicitCommunicationBehavior			
Note	This meta-class represents the ability to define a collection of RunnableEntities. The collection can be nested.			
Base	ARObject, AtpClassifier, AtpFeature, AtpStructureElement, Identifiable , Multilanguage Referrable, Referrable			
Attribute	Datatype	Mul.	Kind	Note
runnableEntity	RunnableEntity	*	iref	This represents a collection of RunnableEntities that belong to the enclosing RunnableEntityGroup. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime
runnableEntityGroup	RunnableEntityGroup	*	iref	This represents the ability to define nested groups of RunnableEntities. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime

Table 4.92: RunnableEntityGroup

Class	DataPrototypeGroup			
Package	M2::AUTOSARTemplates::SWComponentTemplate::ImplicitCommunicationBehavior			
Note	This meta-class represents the ability to define a collection of DataPrototypes that are subject to the formal definition of implicit communication behavior. The definition of the collection can be nested.			
Base	ARObject, AtpClassifier, AtpFeature, AtpStructureElement, Identifiable , Multilanguage Referrable, Referrable			
Attribute	Datatype	Mul.	Kind	Note
dataPrototypeGroup	DataPrototypeGroup	*	iref	This represents the ability to define nested groups of VariableDataPrototypes. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime
implicitDataAccess	VariableDataPrototype	*	iref	This represents a collection of VariableDataPrototypes that belong to the enclosing DataPrototypeGroup Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime

Table 4.93: DataPrototypeGroup

4.9.1 Consistency Needs on Receiver Side

[TPS_SWCT_01472] Receiving [SwComponentType](#) owns a [DataPrototypeGroup](#) in the role [dpgRequiresCoherence](#) [If a receiving [SwComponentType](#) owns a [DataPrototypeGroup](#) in the role [dpgRequiresCoherence](#) for one or several of its [RunnableEntities](#) it is required that [VariableDataPrototypes](#) belonging to the same [DataPrototypeGroup](#) are produced coherently. This means

that the values of the `VariableDataPrototypes` shall be of the same age. `](RS_SWCT_03065)`

[TPS_SWCT_01473] Receiving `SwComponentType` owns a `RunnableEntityGroup` in the role `regRequiresStability` `[` If a receiving `SwComponentType` owns a `RunnableEntityGroup` in the role `regRequiresStability` for one or several of its `RunnableEntities` it is required that the values of implicitly communicated `VariableDataPrototypes` are kept stable over the execution of all `RunnableEntities` belonging to the given `RunnableEntityGroup`. `](RS_SWCT_03065)`

[TPS_SWCT_01474] Receiving `SwComponentType` owns a `RunnableEntityGroup` in the role `regRequiresStability` and also owns one or several `DataPrototypeGroups` in the role `dpgRequiresCoherence` `[` If a receiving `SwComponentType` owns a `RunnableEntityGroup` in the role `regRequiresStability` and also owns one or several `DataPrototypeGroups` in the role `dpgRequiresCoherence` it is required that values of `VariableDataPrototypes` belonging to the same `DataPrototypeGroup` are produced coherently.

This means that the values of the `VariableDataPrototypes` shall be of the same age **and** are kept stable over the execution of all `RunnableEntities` belonging to the given `RunnableEntityGroup`. `](/)`

4.9.2 Consistency Needs on Sender Side

[TPS_SWCT_01475] Sending `SwComponentType` owns a `DataPrototypeGroup` in the role `dpgRequiresCoherence` `[` If a sending `SwComponentType` owns a `DataPrototypeGroup` in the role `dpgRequiresCoherence` for one or several of its `RunnableEntities` it is required that `VariableDataPrototypes` belonging to the same `DataPrototypeGroup` are propagated at the same point of time to `RunnableEntities` which are not belonging to the group of **producing** `RunnableEntities` (which may, but don't have to be formally described as a `RunnableEntityGroup`). `](RS_SWCT_03065)`

The coherence is created at the point in time when the `RunnableEntities` of the producing group of `RunnableEntities` terminate (and the implicit data get updated).

If those `RunnableEntities` are reading the data also, those read accesses will not read the coherent values but the intermediary values written by `RunnableEntities` of the same group.

For all other `RunnableEntities` that are not member of the producing group of `RunnableEntities` it appears as if the data have been updated at this very point coherently.

In order to avoid incorrect configurations its possible to explicitly define the group of `RunnableEntities` for which the coherency does not apply.

[TPS_SWCT_01625] Sending `SwComponentType` owns a `DataPrototypeGroup` in the role `dpgRequiresCoherence` and also `RunnableEntityGroups` `[` If

a sending `SwComponentType` owns a `DataPrototypeGroup` in the role `dp-gRequiresCoherence`, `RunnableEntityGroups` in the role `regDoesNotRequireStability` may exist.

Read accesses from `RunnableEntity`s in those `RunnableEntityGroups` will not read the coherent values but the intermediary values written by `RunnableEntity`s of the same group. [|\(RS_SWCT_03065\)](#)

4.9.3 Consistency Needs for Senders and receivers of the same Data inside on `RunnableEntityGroup`

[TPS_SWCT_01476] Sender and receiver of the same implicitly communicated `VariableDataPrototypes` are associated with the same `RunnableEntityGroup` [|](#) For the case of sender and receiver of the same implicitly communicated `VariableDataPrototypes` are associated with the same `RunnableEntityGroup` [\[TPS_SWCT_01472\]](#), [\[TPS_SWCT_01473\]](#), [\[TPS_SWCT_01475\]](#) as well as [\[TPS_SWCT_01475\]](#) apply with the exception that updates of the values of implicitly communicated `VariableDataPrototypes` inside the given `RunnableEntityGroup` become visible **immediately** after the producing `RunnableEntity` was terminated. [|\(RS_SWCT_03065\)](#)

5 Data Description

5.1 Introduction

[TPS_SWCT_01229] **Three different levels of abstraction regarding the definition of data types** [In the context of defining data types and prototypes, the AUTOSAR concept distinguishes between three different levels of abstraction as depicted in Table 5.1.] ([RS_SWCT_03215](#), [RS_SWCT_03216](#), [RS_SWCT_03217](#))

Application Data Level
Implementation Data Level
Base Type Level

Table 5.1: Abstraction Levels for Describing Data

[TPS_SWCT_01230] **Application Data Level** [The **Application Data Level** is the common level at which [ApplicationSwComponentTypes](#) specify a data type or prototype. This level allows to define all the data attributes which are needed from the application point of view, in order to exchange data between software components or between a software component and a measurement and calibration tool. It is possible to specify data communication of a complete [Virtual Function Bus](#) based on this level only.

This level includes among other things the numerical range of values, the data structure as well as the physical semantics. Data semantics (e.g. physical units) is not in the focus¹ for the RTE in order to make communication technically possible. However, it is important for a unique interpretation of data in the application software and in measurement and calibration systems.] ([RS_SWCT_03216](#))

In former version of this specification, this level was not clearly separated from the implementation level. These had the following drawbacks which are now solved:

- The model of primitive types (like integer, boolean, real, opaque) was anticipating implementation aspects already on a very high level of design.
- The data type model used within ports, focusing on communication via the RTE, was not sufficient to model all type-aspects of variables and parameters which are visible within an AUTOSAR system for other purposes than RTE-communication, namely NvM-data access, calibration, measurement, diagnostics, BSW-module interfaces. Using a uniform type system covering all these aspects is now favored.

¹There are some aspects that affect the RTE, e.g. scaling of [dataElements](#)

- Calibration parameters were not completely incorporated into the data type concept. Some of their attributes (especially for curves and maps) could be specified only on the level of prototypes or were not completely formalized within AUTOSAR (like [SwRecordLayout](#)).
- The data type system was not compatible with the usage in calibration standards like ASAM-MCD (namely the usage of [categorys](#)).
- Adding implementation specific elements like a base type, was not possible without formally changing the data type used in a VFB design. A mapping mechanism that could be used in later project phases and is common in other parts of AUTOSAR (e.g. for mapping components to ECUs) was missing.
- The RTE Specification contained many default rules and assumptions on how to implement certain data types or prototypes in C. With a more formal description of all relevant implementation aspects, the generation of C-interfaces is better determined. But these aspects should be separated from the application level design.
- Since there could be many data types on the application level in a big system, the probability of name clashes in the interfaces to the RTE was rather high. Using a separate set of types to implement the RTE interfaces solves this issue.

[TPS_SWCT_01231] Application level may impose strong requirements on the design of the corresponding implementation level [It should be pointed out, that with the specification of computation methods and record layouts, the application level imposes strong requirements on the design of the corresponding implementation level (for further information see [6.2.5](#)). It might even be the case, that when anticipating different implementations, these elements might be chosen differently.

This is due to the nature of these elements which form a bridge from the physical world to the numerical representation (and vice versa). Nonetheless we consider the specification of these elements as belonging to the application level.

On the one hand, this information is required by MCD-tools and thus shall be part of a rather high-level design. On the other hand, this approach will allow to use a limited set of implementation data types.]([RS_SWCT_03215](#), [RS_SWCT_03216](#), [RS_SWCT_03217](#))

[TPS_SWCT_01232] Implementation Data Level [The *Implementation Data Level* is closer to the actual code implementation in a programming language like C, though it is still an abstraction of the code.

Its values correspond to the actual binary numbers handled by the programming language on the CPU. It contains concepts like pointers and unions which relate to the organization of data in memory and are not relevant for the application level.

This level also defines structure, but it can be more granular. For example, the application level may define a text to be transferred to an instrument cluster as a primitive type (if the structure is not relevant for the application), whereas on the implementation level it could be modeled as an array of bytes.]([RS_SWCT_03217](#))

[TPS_SWCT_01233] Use case for the Implementation Data Level [There are several use cases for this level in AUTOSAR:

- First of all, the *Implementation Data* level can be used in the description of interfaces, and data (e.g. debug data) within the basic software, see [7] for more details on these use cases.
- [ImplementationDataTypes](#) should also be used to describe the interfaces of libraries which operate on a purely numerical level.
- *Implementation Data* is also used for the description of interfaces between software-components and the basic software (namely AUTOSAR Services), because these typically cover implementation aspects only.
- It is possible to define communication in a VFB system directly on this level if the physical and semantical abstraction is not of interest.
- Last not least the input for the RTE generator is defined by data descriptions on this level. This means that in case a SWC defines its data only on application level a corresponding set of implementation data types shall be created (or generated) as part of the ECU extract before the RTE can be generated.

]([RS_SWCT_03217](#))

[TPS_SWCT_01234] Base Level [The ***Base Type Level*** is used to describe the primitive elements in terms of bits and bytes from which the implementation data is built up. It is considered as a separate level in order to allow for reuse of the basic types defined on this level.

These base types still do not completely determine the actual implementation on a programming language, but they impose strong restrictions for this as they define for example the number of bits and bytes to be used.

Depending on the use case, the base types can be defined as platform independent or can also contain platform specific attributes (namely endianness and alignment).]()

[TPS_SWCT_01235] Mapping of data defined on the *Application* level to the *Implementation* and *Base Type* level [It is important to understand, that the mapping of data defined on the *Application* level to the *Implementation* and *Base Type* level depends on the medium on which the data is transported.

For example, if a physical value can be expressed with sufficient accuracy and range by a 16-bit unsigned integer, it still might look very different when sent over CAN, when seen by a software-component on a *big-endian* 32-bit machine or when seen by a software-component on a *little-endian* 16-bit processor.

Conversion between several data implementations of the same application data type might be necessary in case of communication between components on different ECUs. AUTOSAR COM [21] is responsible for this.

It implies that the configuration depends on the definition of the data that are transmitted between components².]([RS_SWCT_03215](#), [RS_SWCT_03216](#), [RS_SWCT_03217](#))

AUTOSAR COM might need to convert a 16-bit integer between *little-endian* and *big-endian* representations; whereas an array of 16 bytes does not need to be swapped even if the endianness changes. In case of intra-ECU communication byte order conversion is not necessary, since the software-components reside on the same machine.

[TPS_SWCT_01236] Big picture of data types [Another way of approaching the concept of data types in AUTOSAR (especially with respect to the question of what “kind” of data type in related to which modeling meta-level) is to sketch the following “big picture” of data types:

ApplicationDataType Defined on **M2** - provides the meta model for data types on application level. It covers the application-relevant aspects of a data type.

An [ApplicationDataType](#) shall finally be mapped to an [ImplementationDataType](#).

ImplementationDataType Defined on **M2** - provides the meta-model for data types on implementation level. With respect to C source code, an [ImplementationDataType](#) finally boils down to a `typedef`.

BaseType Defined on **M2** - provides the platform-dependent part of an [ImplementationDataType](#). the dependency on the platform covers the following aspects:

- Definition on the level of the C language - using [nativeDeclaration](#)
- Technical representation on the target platform (byte order, alignment, encoding) as required for the support of MCD systems.

Platform Data Type Defined on **M1** - provided by AUTOSAR. Platform types shall be available on each platform on which an AUTOSAR-System can run.

The name of the `Platform Data Type` and the properties with respect to the interface between modules / components is the same on every platform.

The particular representation varies from platform to platform.

`Platform Data Types` shall be **modeled** using [ImplementationDataTypes](#).

Note that in AUTOSAR R3.x the platform types are implemented manually and could even not be expressed on ARXML model (see [SRS_Rte_00150]). In AUTOSAR R4.1 the `Platform Data Types` can be represented in the ARXML model. Subsequent releases of AUTOSAR may generate the `Platform Data Types` directly from the ARXML Model.

Standard Type Defined on **M1** - provided by AUTOSAR. Standard types are defined by referring to platform types.

²More exactly speaking, the data shall be converted to and from a so-called [SystemSignal](#).

⌋([RS_SWCT_03215](#), [RS_SWCT_03216](#), [RS_SWCT_03217](#))

[TPS_SWCT_01237] [SwDataDefProps](#) ⌈ The properties of data are summarized in the meta-class [SwDataDefProps](#). This meta-class itself is the superset of all applicable properties. ⌋([RS_SWCT_03216](#), [RS_SWCT_03217](#))

Subsets of [SwDataDefProps](#) are applicable in specific case, for a summary please refer to the following tables:

- The data [categorys](#) are summarized in table [5.7](#).
- Properties for [ApplicationDataTypes](#) are summarized in table [5.8](#).
- Properties for [ImplementationDataTypes](#) are summarized in table [5.18](#).
- Properties for [DataPrototypes](#) typed by [ApplicationDataTypes](#) are summarized in table [5.31](#).
- Properties for [DataPrototypes](#) typed by [ImplementationDataTypes](#) are summarized in table [5.32](#).
- Applicability of [SwDataDefProps](#) is summarized in table [5.39](#).

5.2 Data Types

5.2.1 Overview

As explained in section [5.1](#) it is possible to describe data provided by a software-component from the application as well as from the implementation point of view.

[TPS_SWCT_01072] [ApplicationDataType](#) and [ImplementationDataType](#) ⌈ The common concept behind this is expressed by the abstract meta-class [AutosarDataType](#), from which an [ApplicationDataType](#) and an [ImplementationDataType](#) is derived. ⌋([RS_SWCT_03215](#), [RS_SWCT_03216](#), [RS_SWCT_03217](#))

Figure [5.1](#) shows a summary of the basic meta-classes used for the definition of [AutosarDataTypes](#).

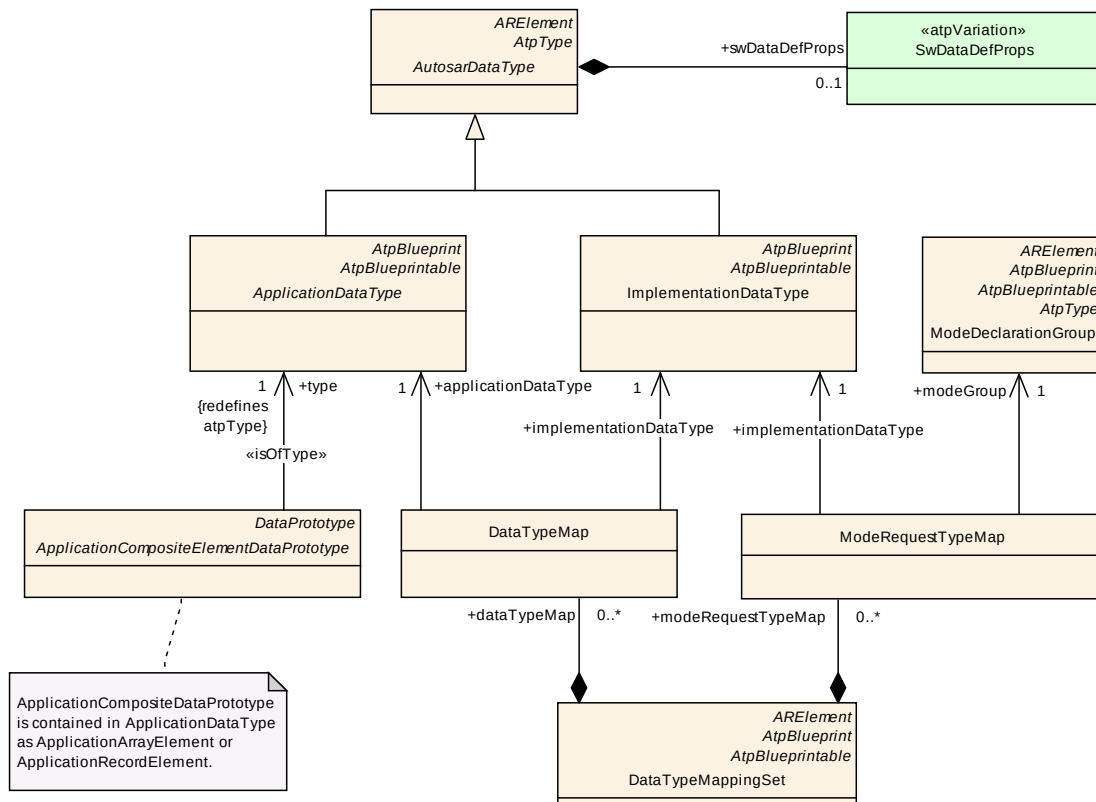


Figure 5.1: Summary of **AutosarDataType**

Class	AutosarDataType (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::Datatypes			
Note	Abstract base class for user defined AUTOSAR data types for ECU software.			
Base	ARElement, ARObjct, AtpClassifier, AtpType, Collectable Element, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Attribute	Datatype	Mul.	Kind	Note
swDataDef Props	SwDataDefProps	0..1	aggr	The properties of this AutosarDataType.

Table 5.2: AutosarDataType

Class	ApplicationDataType (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::Datatypes			
Note	ApplicationDataType defines a data type from the application point of view. Especially it should be used whenever something "physical" is at stake. An ApplicationDataType represents a set of values as seen in the application model, such as measurement units. It does not consider implementation details such as bit-size, endianness, etc. It should be possible to model the application level aspects of a VFB system by using ApplicationDataTypes only.			
Base	ARElement , ARObject , AtpBlueprint , AtpBlueprintable , AtpClassifier , AtpType , Autosar DataType , CollectableElement , Identifiable , MultilanguageReferrable , PackageableElement , Referrable			
Attribute	Datatype	Mul.	Kind	Note
—	—	—	—	—

Table 5.3: ApplicationDataType

Class	ImplementationDataType			
Package	M2::AUTOSARTemplates::CommonStructure::ImplementationDataTypes			
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 , AtpBlueprint , AtpBlueprintable , AtpClassifier , AtpType , Autosar DataType , CollectableElement , Identifiable , MultilanguageReferrable , PackageableElement , Referrable			
Attribute	Datatype	Mul.	Kind	Note
dynamicArraySizeProfile	String	0..1	attr	Specifies the profile which the array will follow in case this data type is a variable size array.
subElement (ordered)	ImplementationDataTypeElement	*	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 ImplementationDataType. Stereotypes: atpSplitable Tags: atp.Splitkey=shortName
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 5.4: ImplementationDataType

[TPS_SWCT_01073] Composite [ApplicationDataType](#) [An [ApplicationDataType](#) can be composed (in form of a record or an array) of elements which themselves are typed by another [ApplicationDataType](#).]([RS_SWCT_03215](#), [RS_SWCT_03216](#))

h [TPS_SWCT_01074] Composite [ImplementationDataType](#) [An [ImplementationDataType](#) can also be composed of elements but in this case no type/prototype concept (see [12]) has been applied. Both concepts will be explained in the following chapters in more detail.]([RS_SWCT_03215](#), [RS_SWCT_03217](#))

5.2.2 Data Type Mapping

As explained above, the concept of application data types as well as that of implementation data types can be used to instantiate a data prototype in an M1 model. However there are use cases, especially in order to generate the RTE contract for [ApplicationSwComponentTypes](#), where it is required to consider both levels for one given data prototype.

[TPS_SWCT_01189] [DataTypeMap](#) [This is supported by the meta-class [DataTypeMap](#) by which an [ApplicationDataType](#) and an [ImplementationDataType](#) can be mapped to each others in order to describe both aspects of one [dataElement](#).]([RS_SWCT_03216](#), [RS_SWCT_03217](#), [RS_SWCT_03215](#))

Class	DataTypeMap			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::Datatypes			
Note	This class represents the relationship between ApplicationDataType and its implementing ImplementationDataType .			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
application DataType	ApplicationData Type	1	ref	This is the corresponding ApplicationDataType
implement ationDataT ype	Implementation DataType	1	ref	This is the corresponding ImplementationDataType .

Table 5.5: [DataTypeMap](#)

If, for example, a [dataElement](#) in a [SenderReceiverInterface](#) is typed by an [ApplicationDataType](#) it shall additionally be associated to an [ImplementationDataType](#) in order to be able to generate the RTE.

[TPS_SWCT_01190] [ModeRequestTypeMap](#) [Another mapping class, [ModeRequestTypeMap](#), has been introduced in order to allow the transport of mode related information via “normal” sender-receiver communication. Apart from this, mode information is not handled by the usual type system but needs special meta-classes. This is explained in more detail in chapter [4.2.5](#).]([RS_SWCT_03110](#))

Note that the mapping classes instead of direct associations have been introduced for process reasons: It allows to maintain application and implementation types in separate M1 artifacts without direct links.

For example, if a software component is moved to another hardware platform the mapping between application and implementation types might be changed in the scope of the specific component without changing the overall VFB model.

[TPS_SWCT_01191] mapped [ApplicationDataType](#) and [ImplementationDataType](#) shall be compatible [In order to set up a valid [DataTypeMap](#) between an [ApplicationDataType](#) and an [ImplementationDataType](#) the two types shall be compatible.

This is further explained in chapter 6.2.5. Of course, if [ImplementationDataTypes](#) are generated from existing [ApplicationDataTypes](#) it is expected that they will be automatically compatible.]([RS_SWCT_03216](#), [RS_SWCT_03217](#))

Furthermore, the various mappings are aggregated in a container [DataTypeMappingSet](#) for easier maintenance in artifacts.

Class	DataTypeMappingSet			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::Datatypes			
Note	This class represents a list of mappings between ApplicationDataTypes and ImplementationDataTypes. In addition, it can contain mappings between ImplementationDataTypes and ModeDeclarationGroups. Tags: atp.recommendedPackage=DataTypeMappingSets			
Base	ARElement , ARObject , AtpBlueprint , AtpBlueprintable , CollectableElement , Identifiable , MultilanguageReferrable , PackageableElement , Referrable			
Attribute	Datatype	Mul.	Kind	Note
dataTypeMap	DataTypeMap	*	aggr	This is one particular association between an ApplicationDataType and its ImplementationDataType .
modeRequestTypeMap	ModeRequestTypeMap	*	aggr	This is one particular association between an ModeDeclarationGroup and its ImplementationDataType .

Table 5.6: [DataTypeMappingSet](#)

Note that the meta-classes [AutosarDataType](#), [ModeDeclarationGroup](#) and [DataTypeMappingSet](#) are derived from [ARElement](#). This means that these and the meta-classes derived from them can be declared on the M1 level as part of an [ARPackage](#) and thus can be used in several different Software Component or Basic Software Module Descriptions.

How to organize [DataTypeMappingSets](#) for a software system, for example whether there is a separate mapping set for each ECU or even for each software component, is considered as project specific. However, the RTE generator needs a well defined [DataTypeMappingSet](#) as input in relation those artifacts which might define data typed as [ApplicationDataTypes](#).

[TPS_SWCT_01192] Meta-classes that have an association to a [DataTypeMappingSet](#) [Therefore, the following meta-classes in the scope of this document have an association to a [DataTypeMappingSet](#):

- [InternalBehavior](#), because it represents the interface between the software component's code and the RTE and all data types belonging to the particular component type have to be uniquely provided on implementation level.
- [ParameterSwComponentType](#), for the same reason (this component type doesn't have an [InternalBehavior](#)).
- [NvBlockDescriptor](#), because this meta-class also leads to generation of code from data types and is not associated to an [InternalBehavior](#).
- [CompositionSwComponentType](#), to support the definition of [ComSpecs](#) in the context of a [CompositionSwComponentType](#). Please note that this definition of a data type mapping is informal (i.e. it shall be taken as a hint for delegation [PortPrototypes](#) that are not yet referenced by a [DelegationSwConnector](#) or [PassThroughSwConnector](#)) and shall **not** be regarded as a binding contract towards the inner elements of the [CompositionSwComponentType](#).

]()

For more details about this aspect please refer to figure [5.60](#).

[TPS_SWCT_01193] Mappings between application and implementation types do not necessarily have to form a 1:1 relation [In general, it is not required that the sum of all mappings between [ApplicationDataType](#) and [ImplementationDataType](#) in a given system form a 1:1 relation. Depending on the use case and on the scope, 1:n as well as n:1 mappings are possible:

- Several different [ApplicationDataTypes](#) may be mapped to the same [ImplementationDataType](#) in the scope of a system, an ECU, or even a single [InternalBehavior](#) of an atomic software component.

Of course, this requires that the different [ApplicationDataTypes](#) are used for different [DataPrototypes](#) and thus that the [DataPrototypes](#) are typed by them (and not by the [ImplementationDataTypes](#)). This allows to establish a more simple type system on the implementation level, than on the application model level.

- The same [ApplicationDataTypes](#) may be mapped to different [ImplementationDataTypes](#) for different ECUs. This scenario allows to chose the implementation data types according to the needs of specific ECUs.
- **[constr_1004] Mapping of [ApplicationDataTypes](#)** [The same [ApplicationDataTypes](#) may be mapped to different [ImplementationDataTypes](#) even in the scope of a single ECU (more exactly speaking, a single RTE), but not in the scope of a single atomic software component.]()

This improves the portability of software components which were developed independently or are ported between ECUs.

]()

[constr_1005] Compatibility of `ImplementationDataTypes` mapped to the same `ApplicationDataType` [It is required that `ImplementationDataTypes` which are taken for connecting corresponding elements of `PortInterfaces` and thus refer to compatible `ApplicationDataTypes` are also compatible among each other (so that RTE is able to cope with possible connections by converting the data accordingly).]()

This constraint is visualized in figure 5.2.

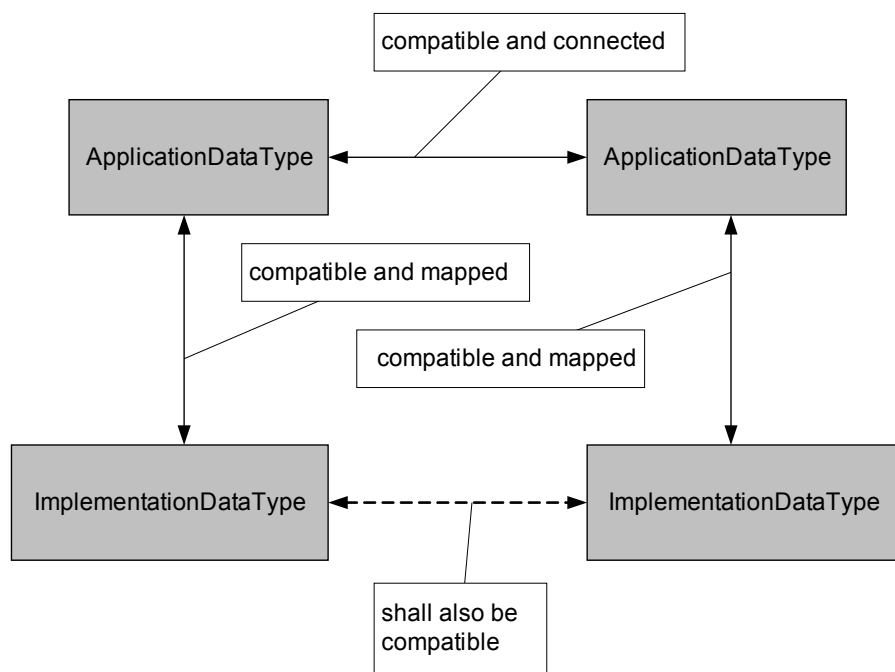


Figure 5.2: Compatibility of Data Types

5.2.3 Data Categories

An `AutosarDataType` is derived from `Identifiable`, thus having a `longName`, a `shortName`, a `category`, and several further attributes for administrative and documentation purposes (for details see [12]).

[TPS_SWCT_01238] Attribute `category` used in the context of `Autosar-DataType` [The `category` attribute is used to set constraints for the various properties which can be specified for an `AutosarDataType`. These properties are defined

by aggregating the meta-class `SwDataDefProps` which contains several attributes and references, see detailed description in chapter 5.4 and 5.4. `()`

[constr_1143] category of `AutosarDataType` shall not be extended `()` In contrast to the general rule that `category` can be extended by user-specific values it is **not allowed** to extend the meaning of the attribute `category` of meta-class `Autosar-DataType` `()`

This approach avoids a very deep and complicated inheritance tree which otherwise would be needed on the M2 level for `AutosarDataType`. There is to some extent a redundancy between setting the `category` and defining the attributes of `Autosar-DataType.swDataDefProps`. This redundancy is intended and allows to for a tool to rule out senseless configurations via simple rules.

In former version of this specification the categories were only used for calibration parameters. Due to several extensions the categories are now applicable for all use cases of the `AutosarDataType`.

An overview on all valid `category`s defined for `AutosarDataType` is shown in table 5.7. Some of the `category`s are also applied to sub-elements of the type system (column “Applicable to ...” in table 5.7). This is explained in more detail in the following sections.

Please note that the column “RTE + BSW” of table 5.7 is only applicable for `category`s that are relevant either for `ImplementationDataTypes` and/or the aspect of measurement and calibration in `McDataInstance`.

[constr_1006] applicable data categories `()` Table 5.7 defines the applicable `category`s depending on specific model elements related to data definition properties. `()`

Category	Applicable to ...											Use Case	Description
	<code>ApplicationArrayDataType</code>	<code>ApplicationRecordDataType</code>	<code>ApplicationPrimitiveDataType</code>	<code>ApplicationRecordElement</code>	<code>ApplicationArrayElement</code>	<code>ApplicationValueSpecification</code>	<code>ImplementationDataType</code>	<code>ImplementationDataTypeElement</code>	<code>SwServiceArg</code>	<code>SwSystemconst</code>	<code>McDataInstance</code>	<code>Calibration</code>	
												<code>Measurement</code>	
												<code>Communication Port Interfaces</code>	
												<code>RTE + BSW</code>	
VALUE			x	x	x	x	x	x		x	x	x	Contains a single value.
VAL_BLK			x	x	x	x				x	x		A value block defines values stored together within one calibration parameter object. It is similar to an value array but it stores the values by means of an axis instead (only important for calibration data handling).
DATA_REFERENCE						x	x	x				x	Contains an address of another <code>DataPrototype</code> (whose type is given via <code>SwDataDefProps.swPointerTargetProps</code>).
FUNCTION_REFERENCE						x	x	x				x	Contains an address of a function prototype (whose signature is given via <code>SwDataDefProps.swPointerTargetProps.functionPointerSignature</code>).

Category	Applicable to ...											Use Case		Description		
	ApplicationArrayDataType	ApplicationRecordDataType	ApplicationPrimitiveDataType	ApplicationRecordElement	ApplicationArrayElement	ApplicationValueSpecification	ImplementationDataType	ImplementationDataTypeElement	SwServiceArg	SwSystemconst	McDataInstance	Calibration	Measurement	Communication Port Interfaces	RTE + BSW	
TYPE_REFERENCE						x	x	x					x	x		The element is defined via reference to another data type (via SwDataDefProps.implementationDataType).
STRUCTURE	x			x	x		x	x			x	x	x	x	x	Holds one or several further elements which can have different AutosarDataTypes . The underlying elements are defined in the same manner as normal data except for the association to SwAddrMethod : This has to be the same for all underlying elements. Corresponds to a Record if used in the application domain.
UNION							x	x			x	x	x		x	Can hold values of different data types. It is similar to STRUCTURE except that all of its members start at the same location in memory. A UNION data prototype can contain only one of its elements at a time. The size of the UNION is at least the size of the largest member.
ARRAY	x			x	x		x	x			x	x	x	x	x	An array of sub-elements which are of the same type.
BIT											x	x	x		x	One or several bits within a host variable, which are treated as an own data object.
HOST											x	x	x		x	A HOST data type is like a simple VALUE , but it is used for packed bit definition. That means it can host several BIT variables which have their own description and measurement access.
STRING			x	x	x	x					x	x	x	x		Contains a single value interpreted as a text string (note that it appears as a single value for the application domain; the internal representation can be an array).
BOOLEAN			x	x	x	x					x	x	x	x		Contains one boolean state. Depending on the CPU direct addressing of single bits may not be available. So a byte or a word can be used to store only one logical state.
COM_AXIS			x		x	x					x	x				An axis definition as separate calibration parameter which can be referenced by any CURVE , MAP , CUBOID , CUBE_4 , and CUBE_5 . The benefits by using a common axis is that it saves memory space; because it is stored only one time and can be used in multiple CURVES , MAPS , CUBOIDS , CUBE_4s , and CUBE_5s .
RES_AXIS			x		x	x					x	x				A RES_AXIS (rescale axis) is also a shared axis like COM_AXIS , the difference is that this kind of axis can be used for rescaling. Note that the RES_AXIS is by nature a CURVE which is used to implement a non linear scaling (rescale) of the axis. In addition to saving memory space via the shared usage like a COM_AXIS , it can compress a huge range to a non-linear distributed axis points thus retaining the required accuracy.
CURVE_AXIS			x	x	x	x					x	x				CURVE_AXIS uses a separate CURVE to rescale the axis. The referenced CURVE is used to lookup an axis index, and the index value is used by the controller to determine the operating point in the CURVE , MAP , CUBOID , CUBE_4 , or CUBE_5 .

Category	Applicable to ...										Use Case		Description			
	ApplicationArrayDataType	ApplicationRecordDataType	ApplicationPrimitiveDataType	ApplicationRecordElement	ApplicationArrayElement	ApplicationValueSpecification	ImplementationDataType	ImplementationDataTypeElement	SwServiceArg	SwSystemconst	McDataInstance	Calibration	Measurement	Communication Port Interfaces	RTE + BSW	
CURVE			x	x	x	x					x	x				Calibration parameter with one input value and one output value. That means output values can be defined depending on the input value. The granularity of implemented functionality can be changed by using different number of axis points. A CURVE has always one input axis and one output axis. The output axis is a characteristic of the curve and every time present but the input axis can be defined within the curve definition or separately.
MAP			x	x	x	x					x	x				Calibration parameter with two input values and one output value. That means output values can be defined depending on the input values. The granularity of implemented functionality can be changed by using different number of axis points for y- and x-axis. A MAP has always two input axes and one output axis. The output axis is a characteristic of the MAP and every time present but the input axes can be defined within the MAP definition or separately.
CUBOID			x	x	x	x					x	x				Calibration parameter with three input values and one output value. That means output values can be defined depending on the input values. The granularity of implemented functionality can be changed by using different number of axis points for the input axes. A CUBOID has always three input axes and one output axis. The output axis is a characteristic of the CUBOID and every time present but the input axes can be defined within the CUBOID definition or separately.
CUBE_4			x	x	x	x					x	x				Calibration parameter with four input values and one output value. That means output values can be defined depending on the input values. The granularity of implemented functionality can be changed by using different number of axis points for the input axes. A CUBE_4 has always four input axes and one output axis. The output axis is a characteristic of the CUBE_4 and every time present but the input axes can be defined within the CUBE_4 definition or separately.
CUBE_5			x	x	x	x					x	x				Calibration parameter with five input values and one output value. That means output values can be defined depending on the input values. The granularity of implemented functionality can be changed by using different number of axis points for the input axes. A CUBE_5 has always five input axes and one output axis. The output axis is a characteristic of the CUBE_5 and every time present but the input axes can be defined within the CUBE_5 definition or separately.
MACRO									x					x	x	This represents an argument to a C macro.

Table 5.7: Usage of **category** for Data Types

[TPS_SWCT_01239] default value for attribute **category** used in the context of **SwSystemconst** [The default value for the **category** of a **SwSystemconst** shall be **VALUE**. This has to be applied if no explicit definition of the **category** can be found.]()

5.2.4 Application Data Type

[TPS_SWCT_01240] Subclasses of **ApplicationDataType** [As figure 5.3 explains, the abstract meta-class **ApplicationDataType** is further derived into an **ApplicationPrimitiveDataType** and an **ApplicationCompositeDataType** which are further explained in the following sub-chapters.] (RS_SWCT_03216)

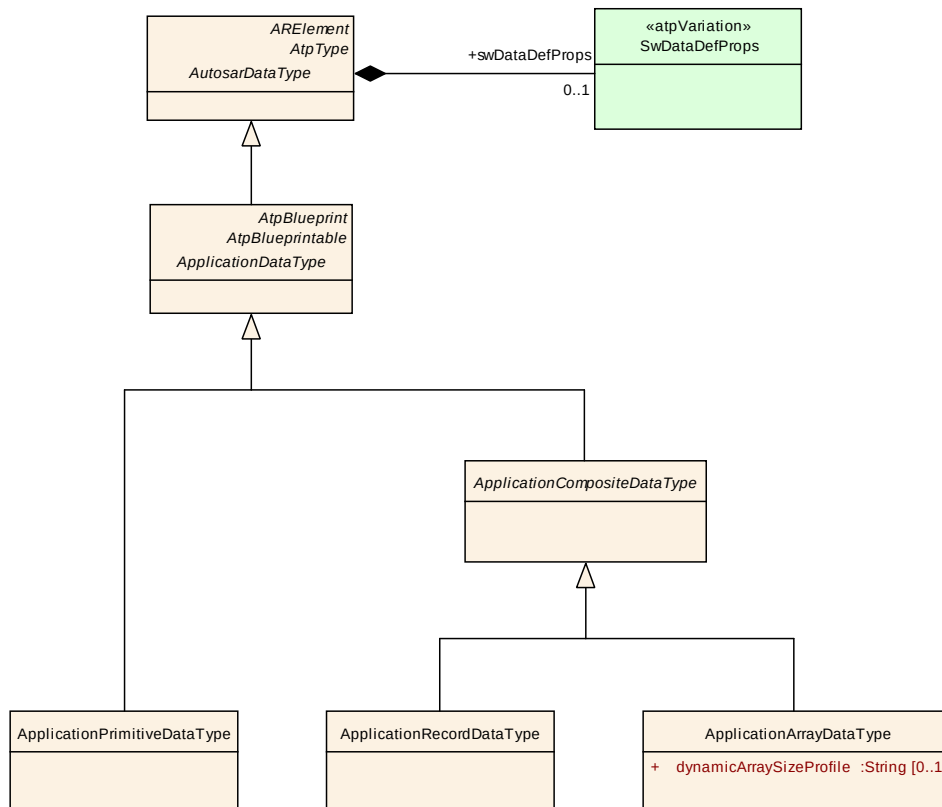


Figure 5.3: Basic Meta-Model for **ApplicationDataType**

	Root Elem.			Attribute Existence per Category												
	ApplicationDataType	ApplicationRecordElement	ApplicationArrayElement	VALUE	VAL_BLK	STRUCTURE	ARRAY	STRING	BOOLEAN	COM_AXIS	RES_AXIS	CURVE	MAP	CUBOID	CUBE_4	CUBE_5
Attributes of SwDataDefProps																
additionalNativeTypeQualifier				*	*	*	*	*	*	*	*	*	*	*	*	*
annotation	X	X	X	*	*	*	*	*	*	*	*	*	*	*	*	*
baseType																
compuMethod	X			0..1	0..1			0..1	0..1			0..1	0..1	0..1	0..1	0..1
dataConstr	X	X	X	0..1	0..1				0..1			0..1	0..1	0..1	0..1	0..1
displayFormat	X	X	X	0..1	0..1			0..1	0..1			0..1	0..1	0..1	0..1	0..1
implementationDataType																
invalidValue	X			0..1				0..1	0..1							
stepSize	X	X	X	0..1	0..1					0..1	0..1	0..1	0..1	0..1	0..1	0..1
swAddrMethod	X			0..1	0..1	0..1	0..1	0..1	0..1	0..1	0..1	0..1	0..1	0..1	0..1	0..1
swAlignment																
swBitRepresentation																
swCalibrationAccess	X			0..1	0..1	0..1	0..1	0..1	0..1	1	1	1	1	1	1	1
swCalprmAxisSet	X									1	1	1	1	1	1	1
swComparisonVariable																
swDataDependency																
swHostVariable																
swImplPolicy	X			0..1	0..1	0..1	0..1	0..1	0..1	0..1	0..1	0..1	0..1	0..1	0..1	0..1
swIntendedResolution	X	X	X	0..1												
swInterpolationMethod	X			0..1						0..1	0..1	0..1	0..1	0..1	0..1	0..1
swIsVirtual																
swPointerTargetProps																
swRecordLayout	X			0..1	0..1 ³			0..1		1	1	1	1	1	1	1
swRefreshTiming	X			0..1	0..1			0..1	0..1							
swTextProps	X							1								
swValueBlockSize	X				1											
unit	X			0..1	0..1			0..1	0..1			0..1	0..1	0..1	0..1	0..1
valueAxisDataType	X				0..1					0..1	0..1	0..1	0..1	0..1	0..1	0..1
Other Attributes below the Root Element																
element: ApplicationRecordElement	X	X	X			1..*										
element: ApplicationArrayElement	X	X	X				1									
ApplicationArrayElement.array- SizeSemantics	X						0..1									
ApplicationArrayElement.maxNum- berOfElements	X						1									

Table 5.8: Allowed Attributes vs. *category* for *ApplicationDataTypes*
³This is required by [TPS_SWCT_01179].

Class	ApplicationPrimitiveDataType			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::Datatypes			
Note	A primitive data type defines a set of allowed values. Tags: atp.recommendedPackage=ApplicationDataTypes			
Base	ARElement , ARObject , ApplicationDataType , AtpBlueprint , AtpBlueprintable , AtpClassifier , AtpType , AutosarDataType , CollectableElement , Identifiable , MultilanguageReferrable , PackageableElement , Referrable			
Attribute	Datatype	Mul.	Kind	Note
—	—	—	—	—

Table 5.9: ApplicationPrimitiveDataType

Class	ApplicationCompositeDataType (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::Datatypes			
Note	Abstract base class for all application data types composed of other data types.			
Base	ARElement , ARObject , ApplicationDataType , AtpBlueprint , AtpBlueprintable , AtpClassifier , AtpType , AutosarDataType , CollectableElement , Identifiable , MultilanguageReferrable , PackageableElement , Referrable			
Attribute	Datatype	Mul.	Kind	Note
—	—	—	—	—

Table 5.10: ApplicationCompositeDataType

[TPS_SWCT_01241] Applicable **categorys** for subclasses of **Application-DataType** [Like any **AutosarDataType**, also the primitive and composite types on application level are characterized by their **category** and their **SwDataDefProps**. For a given **category**, only a limited set of attributes of the **SwDataDefProps** makes sense.]([RS_SWCT_03216](#))

[constr_1007] Allowed attributes of **SwDataDefProps** for **Application-DataTypes** [The allowed attributes of **SwDataDefProps** for **Application-DataTypes** and their allowed multiplicities are listed as an overview in table 5.8.]()

This list makes use of the **SwDataDefProps** and other meta-model elements which are explained in detail in the further sections of this chapter.

[constr_1008] Applicability of **categorys** **STRUCTURE** and **ARRAY** [The categories **STRUCTURE** and **ARRAY** correspond to **ApplicationCompositeDataTypes** whereas all other **categorys** can be applied only for **ApplicationPrimitive-DataTypes**.]()

5.2.4.1 Application Primitive Data Types

5.2.4.1.1 Data Types for Single Values

In contrast to prior versions (R3.x) of the AUTOSAR standard, the primitive application data types on M2 level are no longer specified. Instead of this, the meta-class [ApplicationPrimitiveDataType](#) in combination with the attached [swDataDefProps](#) is used on the level of the M2 (meta-) model to specify the details on M1 modeling level.

[TPS_SWCT_01242] [category](#) characterizes the nature of a data type on application level [The [category](#) is used in addition to characterize the nature of a data type on application level.] ([RS_SWCT_03216](#))

For example, the [IntegerType](#) as of AUTOSAR R3.x allows for specifying lower and upper ranges that constrain the applicable value interval. That aspect is still supported by this version of AUTOSAR, but the meta-model is different from the former approach. Especially it is no more considered of importance to specify that an [ApplicationPrimitiveDataType](#) is actually represented by “integer” numbers.

Figure 5.4 provides a sketch of how limits are defined now. The key feature is the aggregation of [SwDataDefProps](#) at [AutosarDataType](#). The meta-class [SwDataDefProps](#) allows for creating a reference to a [DataConstr](#) that in turn aggregates a [DataConstrRule](#).

The latter aggregates [PhysConstrs](#) and this meta-class finally owns two [Limits](#) in the roles [lowerLimit](#) and [upperLimit](#).

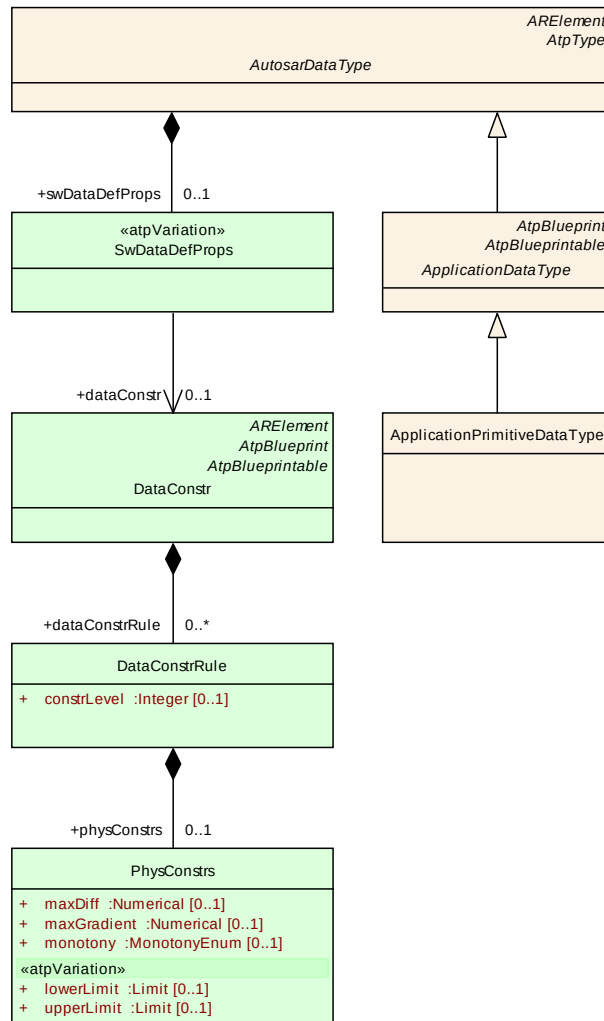


Figure 5.4: Specification of Physical Limits

Another example is shown in Figure 5.5. By making again use of **SwDataDefProps**, this figure shows how semantics in form of a **CompuMethod** and a **Unit** can be attached. Also an **initValue** can be defined which is used by the RTE in order to initialize values of **DataPrototypes** defined locally in a software-component.

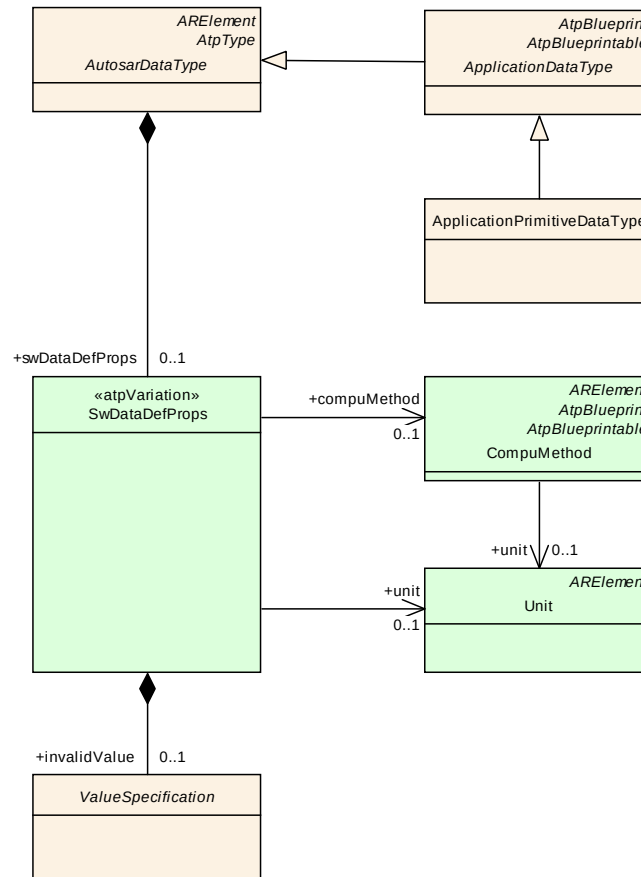


Figure 5.5: Some Properties of [ApplicationPrimitiveDataTypes](#)

Figure 5.6 illustrates the relationship between the data constraints for [Application-DataType](#), [CompuMethod](#), [ImplementationDataType](#), [BaseType](#) and also the [invalidValue](#).

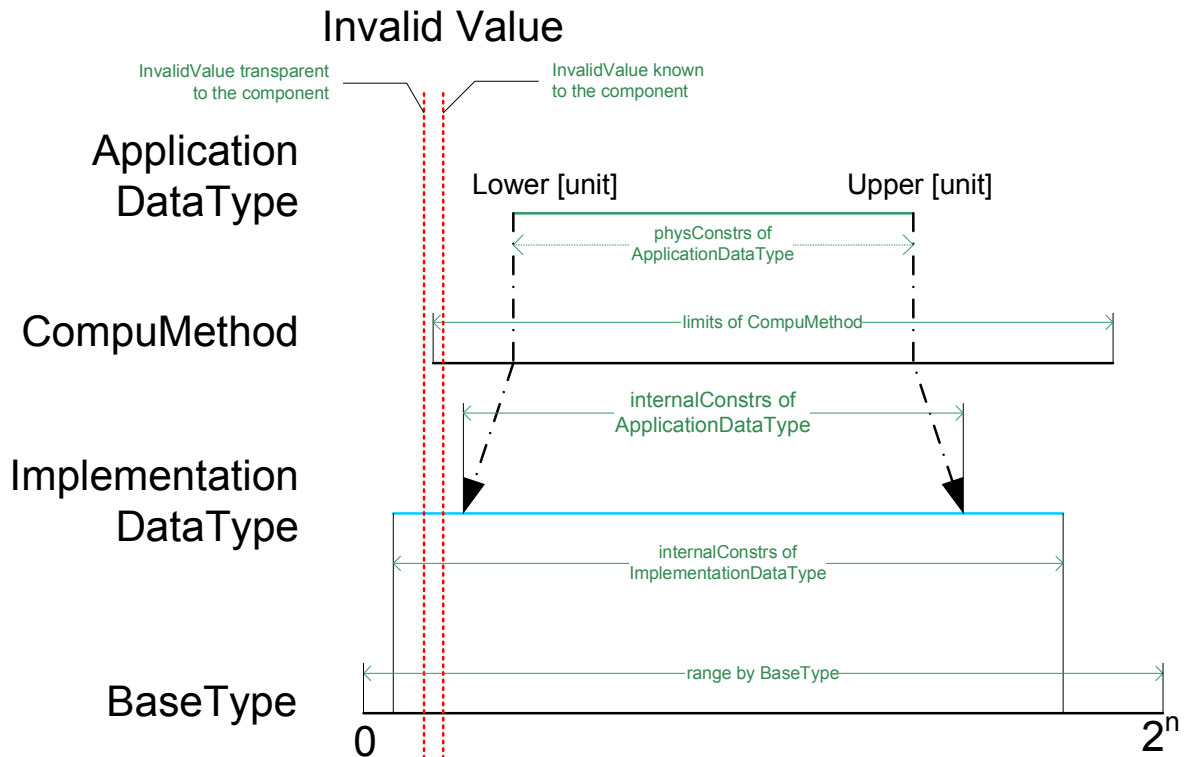


Figure 5.6: Value ranges and invalid values

[constr_2544] Limits need to be consistent [

- The limits of `ApplicationDataType` shall be inside of the definition range of the `CompuMethod`

The `CompuMethod` needs to be applicable for limits of an `ApplicationDataType`. The reason is that the internal representation of the limits for the `ApplicationDataType` are calculated by applying the `CompuMethod`.

- The such defined internal limits of the `ApplicationDataType` shall be within or equal the `internalConstrs` of the mapped `ImplementationDataType`.
- The limits of the `ImplementationDataType` shall be within or equal to the limits defined by the size of the `BaseType`.

]()

[constr_1281] `invalidValue` is inside the scope of the `compuMethod` [If the value of the `invalidValue` of an `ApplicationPrimitiveDataType` of category `VALUE` is supposed to be **inside** the scope of the applicable `CompuMethod` an `ApplicationValueSpecification` is used to describe the `invalidValue` of the `ApplicationPrimitiveDataType`.]()

[constr_1281] means that the value of the `ApplicationValueSpecification` shall be within the bounds defined by `swDataDefProps.compuMethod.compuPhysToInternal.compuContent.compuScale.lowerLimit` resp. `upperLimit` or the inverse case that is based on the bounds defined by `swDataDefProps.com-`

`puMethod.compuInternalToPhys.compuContent.compuScale.lowerLimit`
resp. `upperLimit`.

[constr_1283] `invalidValue` is outside the scope of the `compuMethod` [If the value of the `invalidValue` of an `ApplicationPrimitiveDataType` of category `VALUE` is supposed to be **outside** the scope of the applicable `CompuMethod` a `NumericalValueSpecification` shall be used to describe the `invalidValue` of the `ApplicationPrimitiveDataType`.]()

The handling of `invalidValue` for `ApplicationPrimitiveDataType` of category `STRING` is defined by **[constr_1242]**.

For a more detailed description of the properties that can be defined for data types (and data prototypes as well) see sections 5.4 and 5.4.2.

5.2.4.1.2 About Enumerations

[TPS_SWCT_01243] Definition of enumeration types [In the AUTOSAR meta-model, an enumeration is not implemented by means of an `ApplicationCompositeDataType`.

Instead, a range of integer numbers can be used as a structural description for a single `ApplicationPrimitiveDataType` or an `ImplementationDataType` of category `VALUE` or `TYPE_REFERENCE` that boils down to an `ImplementationDataType` of category `VALUE`.

The mapping of the integer numbers to *labels* in the scope of the definition of an enumeration is considered part of the semantical definition via an attached `CompuMethod` rather than part of the structural description.](*RS_SWCT_03216*)

[TPS_SWCT_01562] Specification of values of an enumeration [For the specification of values of an enumeration on the basis of the labels defined in the applicable `CompuMethod` it is necessary to distinguish two approaches based on the used `AutosarDataType`:

- `ImplementationDataType`: as mentioned by **[constr_1225]**, the definition of the labels of an enumeration shall only be done by using `TextValueSpecification`.
- `ApplicationPrimitiveDataType`: use the `ApplicationValueSpecification.swValueCont.swValuesPhys.vt` or `ApplicationRuleBasedValueSpecification.swValueCont.ruleBasedValues.arguments.vt`.

]()

The relevant meta-classes in the context of `SwDataDefProps` are sketched in Figure 5.7. This includes all meta-classes that may contribute to the definition of the symbol of a `CompuScale` in C code, see **[TPS_SWCT_01431]**.

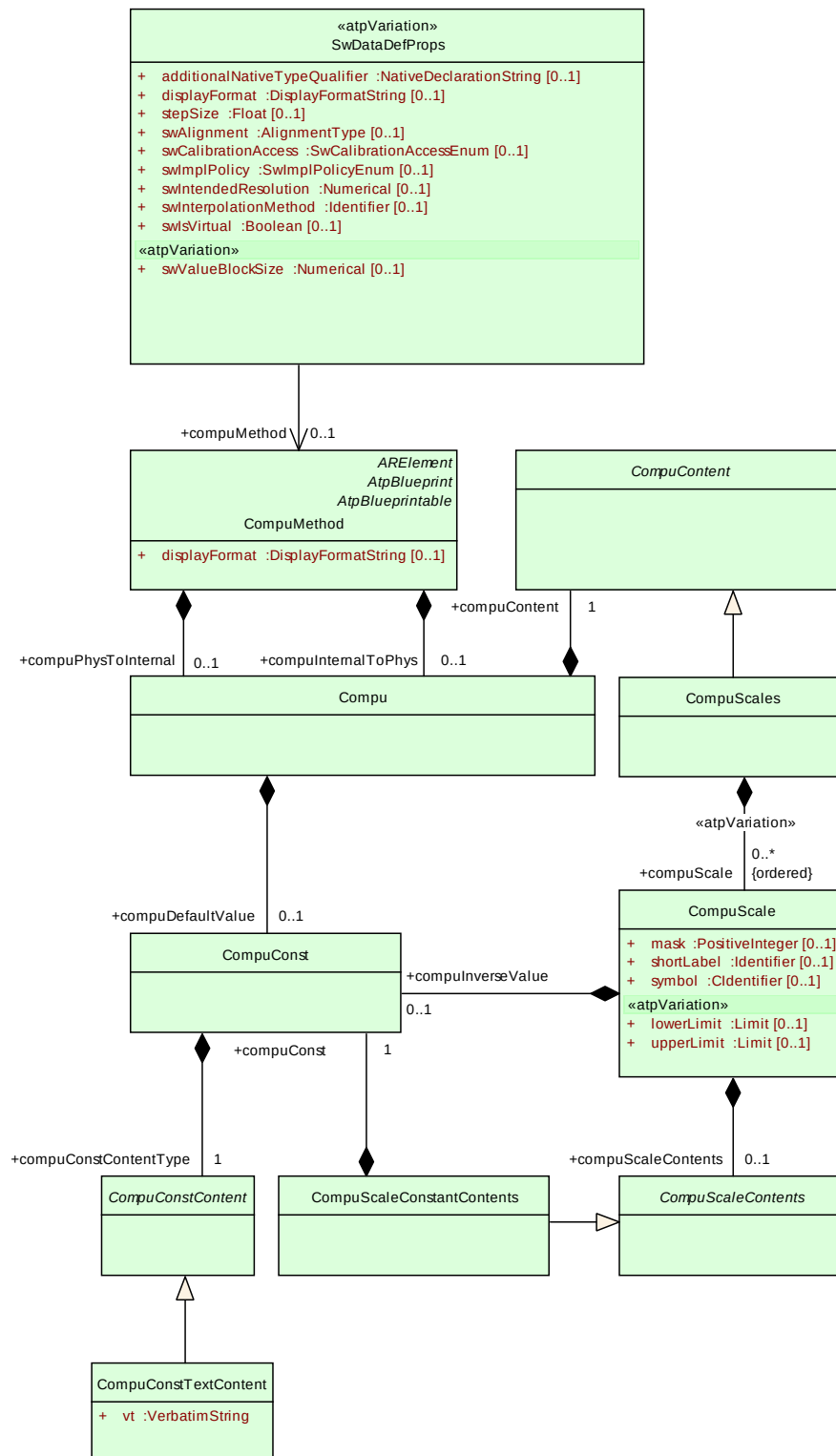


Figure 5.7: Relevant meta-classes for the specification of enumerations

An example of how an enumeration looks like in ARXML is contained in section 5.5.1.3.

5.2.4.1.3 Data Types for Calibration Parameters

[TPS_SWCT_01244] Data types for calibration parameters are also described as primitive types [Data types for calibration parameters are from the application perspective also described as primitive types. This is obvious, if they are simple values (`category VALUE`). Also the `category STRING` is treated as a primitive type on application level.

Less obvious is the fact that `ApplicationDataTypes` of the categories `VAL_BLK`, `COM_AXIS`, `RES_AXIS`, `CURVE`, `MAP`, `CUBOID`, `CUBE_4`, and `CUBE_5` are not described as composite data types (as far as the application level is concerned) although they admittedly possess some kind of internal structure.

In contrast to `ApplicationCompositeDataTypes`, they are **not** composed in a self-similar way of other `AutosarDataTypes`. Their substructure needs a special description in order to be compatible with existing calibration techniques. `]()`

[TPS_SWCT_01245] `SwDataDefProps` control the structure of calibration parameters [The substructure of these types is attached to the `SwDataDefProps`. By this means it is possible to define on the level of `DataPrototypes` or other artifacts, where the `SwDataDefProps` come into play. For details on these part of the `SwDataDefProps` see chapters 5.4.4 and 5.5.5. `]()`

5.2.4.1.4 Data Types for Textual Strings

[constr_1093] Definition of textual strings [An `ApplicationPrimitiveDataType` of `category STRING` shall have a `swTextProps` which determines the `arraySizeSemantics` and `swMaxTextSize`. `]()`

[TPS_SWCT_01488] `ApplicationPrimitiveDataType` shall be interpreted as a string of a particular encoding [To indicate that an `ApplicationPrimitiveDataType` shall be interpreted as a string of a particular encoding it shall reference `swDataDefProps.swTextProps.baseType` and the only attribute of the referenced `SwBaseType` relevant for this purpose is the `BaseTypeDirectDefinition.baseTypeEncoding`. `]()`

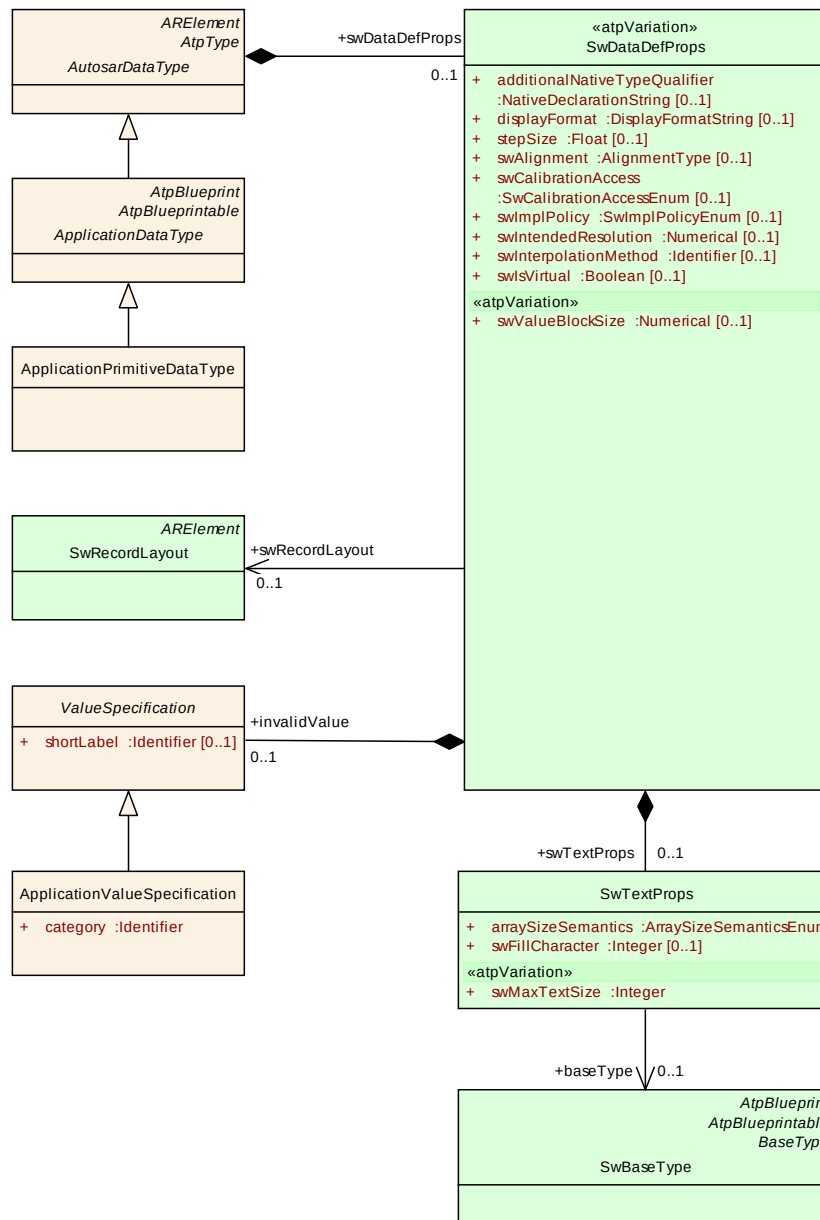


Figure 5.8: Specification of textual strings

Class	SwTextProps			
Package	M2::AUTOSARTemplates::CommonStructure::DataDefProperties			
Note	This meta-class expresses particular properties applicable to strings in variables or calibration parameters.			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note

Attribute	Datatype	Mul.	Kind	Note
arraySizeSemantics	ArraySizeSemanticsEnum	1	attr	This attribute controls the semantics of the arraysize for the array representing the string in an ImplementationDataType. It is there to support a safe conversion between ApplicationDatatype and ImplementationDatatype, even for variable length strings as required e.g. for Support of SAE J1939. In conjunction with swFillCharacter, it provides the following options: - FixedLengthString: FixedSize - no fillcharacter - TerminatedStringFixedLengthCommunication: FixedSize - with fillcharacter - VariableLengthString: VariableSize - no fillcharacter - TerminatedStringVariableLengthCommunication: VariableSize with fillcharacter
baseType	SwBaseType	0..1	ref	This is the base type of one character in the string. In particular this baseType denotes the intended encoding of the characters in the string on level of ApplicationDataType. Tags: xml.sequenceOffset=30
swFillCharacter	Integer	0..1	attr	Filler character for text parameter to pad up to the maximum length swMaxTextSize. The value will be interpreted according to the encoding specified in the associated base type of the data object, e.g. 0x30 (hex) represents the ASCII character zero as filler character and 0 (dec) represents an end of string as filler character. The usage of the fill character depends on the arraySizeSemantics. Tags: xml.sequenceOffset=40
swMaxTextSize	Integer	1	attr	Specifies the maximum text size in characters. Note the size in bytes depends on the encoding in the corresponding baseType. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime xml.sequenceOffset=20

Table 5.11: SwTextProps

[TPS_SWCT_01127] Byte array with variable size [[SwTextProps](#) can be used to define byte arrays of variable size.]([RS_SWCT_03182](#), [RS_SWCT_03181](#))

[TPS_SWCT_01246] [SwRecordLayout](#) may also be required for A2L generation [A [SwRecordLayout](#) may also be required for the generation of A2L if the string is part of calibration data.]()

As stated by [TPS_SWCT_01128], the definition of [SwDataDefProps.swRecordLayout](#) is considered mandatory anyway for [ApplicationPrimitiveDataTypes](#) of category `STRING`.

The following series of XML fragments exemplifies the definition of a data type for the representation of a textual string. First, the applicable [ApplicationPrimitiveDataType](#) is defined (see Figure 5.8):

Listing 5.1: Example for the definition of a string [ApplicationPrimitiveDataType](#)

```
<AR-PACKAGE>
  <SHORT-NAME>ApplicationDataTypes</SHORT-NAME>
  <ELEMENTS>
    <APPLICATION-PRIMITIVE-DATA-TYPE>
      <SHORT-NAME>MyApplicationStringType</SHORT-NAME>
      <CATEGORY>STRING</CATEGORY>
      <SW-DATA-DEF-PROPS>
        <SW-DATA-DEF-PROPS-VARIANTS>
          <SW-DATA-DEF-PROPS-CONDITIONAL>
            <SW-TEXT-PROPS>
              <ARRAY-SIZE-SEMANTICS>VARIABLE-SIZE</ARRAY-SIZE-SEMANTICS>
              <SW-MAX-TEXT-SIZE>50</SW-MAX-TEXT-SIZE>
              <BASE-TYPE-REF DEST="SW-BASE-TYPE" BASE="default">BaseTypes/
                MyTextBaseType</BASE-TYPE-REF>
            </SW-TEXT-PROPS>
            <INVALID-VALUE>
              <APPLICATION-VALUE-SPECIFICATION>
                <CATEGORY>STRING</CATEGORY>
                <SW-VALUE-CONT>
                  <SW-VALUES-PHYS>
                    <VT>inv</VT>
                  </SW-VALUES-PHYS>
                </SW-VALUE-CONT>
              </APPLICATION-VALUE-SPECIFICATION>
            </INVALID-VALUE>
            <SW-RECORD-LAYOUT-REF DEST="SW-RECORD-LAYOUT" BASE="default">
              RecordLayouts/StringDescriptor</SW-RECORD-LAYOUT-REF>
          </SW-DATA-DEF-PROPS-CONDITIONAL>
        </SW-DATA-DEF-PROPS-VARIANTS>
      </SW-DATA-DEF-PROPS>
    </APPLICATION-PRIMITIVE-DATA-TYPE>
  </ELEMENTS>
</AR-PACKAGE>
```

Note that the `category` is set to the value `STRING`. Also the [ApplicationPrimitiveDataType.swDataDefProps.swTextProps](#) indicate the width of the string and also define (by means of the reference to [baseType](#)) the encoding this string data type is supposed to utilize.

Note further that the fact that an `ApplicationDataType` directly references (across the implementation level) to a `SwBaseType` **represents an exception to the rule that `ApplicationDataType` should not be concerned about the lowest level of data type definition** in AUTOSAR.

If the bridging of the implementation level were accepted as a general pattern for the modeling of `ApplicationDataType` it would easily be possible to bypass the implementation level to some extent and this would render `ApplicationDataTypes` less versatile.

[TPS_SWCT_01128] `SwRecordLayout` needed for `ApplicationPrimitiveDataType` of category `STRING` [As mentioned in [TPS_SWCT_01179], an `ApplicationPrimitiveDataType` of category `STRING` is considered a Compound Primitive Data Type.

Therefore, it needs a reference to the definition of a `SwRecordLayout` that presets the approach for creating a matching `ImplementationDataType`.]()

In this specific example the definition of the `SwRecordLayout` foresees the `ApplicationPrimitiveDataType` of category `STRING` to be implemented as a structured data type that consists of:

1. the **size** of an instance of the string data type in terms of the number of characters plus
2. an **array** that can be used to store the individual characters contained in an instance of the string data type.

Depending on the used encoding the **array** may need to be bigger (in terms of the number of elements) than the corresponding value of the **size**. Furthermore, the definition of the `SwRecordLayout` already takes into account that the implementation of an array data type by means of an `ImplementationDataType` requires the definition of an `ImplementationDataTypeElement`.

The meaning of the standardized values of `SwRecordLayoutV.swRecordLayoutVProp` are documented in [TPS_SWCT_01489]. In the scope of this example the values `COUNT` and `VALUE` are used.

The fact that the `swRecordLayoutGroupTo` contains the value -1 means that the iteration ends at the last element of the array.

Listing 5.2: Example for the definition of a `SwRecordLayout` for an `ApplicationPrimitiveDataType` of category `STRING`

```
<AR-PACKAGE>
  <SHORT-NAME>RecordLayouts</SHORT-NAME>
  <ELEMENTS>
    <SW-RECORD-LAYOUT>
      <SHORT-NAME>StringDescriptor</SHORT-NAME>
      <LONG-NAME>
        <L-4 L="EN">String by descriptor</L-4>
      </LONG-NAME>
      <INTRODUCTION>
```



```

<VERBATIM>
  <L-5 L="EN" xml:space="default">
    struct{
      size,
      char[]
    }
  </L-5>
</VERBATIM>
</INTRODUCTION>
<SW-RECORD-LAYOUT-GROUP>
  <SW-RECORD-LAYOUT-V>
    <SHORT-LABEL>size</SHORT-LABEL>
    <SW-RECORD-LAYOUT-V-AXIS>STRING</SW-RECORD-LAYOUT-V-AXIS>
    <SW-RECORD-LAYOUT-V-PROP>COUNT</SW-RECORD-LAYOUT-V-PROP>
  </SW-RECORD-LAYOUT-V>
  <SW-RECORD-LAYOUT-GROUP>
    <SHORT-LABEL>chars</SHORT-LABEL>
    <SW-RECORD-LAYOUT-GROUP-AXIS>STRING</SW-RECORD-LAYOUT-GROUP-AXIS>
    <SW-RECORD-LAYOUT-GROUP-FROM>0</SW-RECORD-LAYOUT-GROUP-FROM>
    <SW-RECORD-LAYOUT-GROUP-TO>-1</SW-RECORD-LAYOUT-GROUP-TO>
    <SW-RECORD-LAYOUT-V>
      <SHORT-LABEL>char</SHORT-LABEL>
      <SW-RECORD-LAYOUT-V-PROP>VALUE</SW-RECORD-LAYOUT-V-PROP>
    </SW-RECORD-LAYOUT-V>
  </SW-RECORD-LAYOUT-GROUP>
</SW-RECORD-LAYOUT-GROUP>
</SW-RECORD-LAYOUT>
</ELEMENTS>
</AR-PACKAGE>

```

Please note further that the discussed example of an [ApplicationPrimitiveDataType](#) of [category](#) `STRING` also contains the definition of an [invalidValue](#) for the string data type.

The next step is the definition of an [ImplementationDataType](#) that represents the string type on the implementation level. The definition of the [ImplementationDataType](#) can be derived from the definition of the applicable [SwRecordLayout](#).

Please note that the [ImplementationDataType](#) **also** defines an [invalidValue](#). As mentioned in [TPS_SWCT_01487], the consistency of the [invalidValue](#) defined in the scope of the [ApplicationPrimitiveDataType](#) of [category](#) `STRING` and the [invalidValue](#) defined in the scope of the corresponding [ImplementationDataType](#) cannot formally be checked.

Listing 5.3: Example for the definition of a string [ImplementationDataType](#)

```

<AR-PACKAGE>
  <SHORT-NAME>ImplementationDataTypes</SHORT-NAME>
  <ELEMENTS>
    <IMPLEMENTATION-DATA-TYPE>
      <SHORT-NAME>uint8</SHORT-NAME>
      <CATEGORY>VALUE</CATEGORY>
      <SW-DATA-DEF-PROPS>
        <SW-DATA-DEF-PROPS-VARIANTS>
          <SW-DATA-DEF-PROPS-CONDITIONAL>

```

```

        <BASE-TYPE-REF DEST="SW-BASE-TYPE">BaseTypes/uint8BaseType</
        BASE-TYPE-REF>
    </SW-DATA-DEF-PROPS-CONDITIONAL>
</SW-DATA-DEF-PROPS-VARIANTS>
</SW-DATA-DEF-PROPS>
</IMPLEMENTATION-DATA-TYPE>
<IMPLEMENTATION-DATA-TYPE>
    <SHORT-NAME>MyImplementationStringType</SHORT-NAME>
    <CATEGORY>STRUCTURE</CATEGORY>
    <SUB-ELEMENTS>
        <IMPLEMENTATION-DATA-TYPE-ELEMENT>
            <SHORT-NAME>size</SHORT-NAME>
            <CATEGORY>TYPE_REFERENCE</CATEGORY>
            <SW-DATA-DEF-PROPS>
                <SW-DATA-DEF-PROPS-VARIANTS>
                    <SW-DATA-DEF-PROPS-CONDITIONAL>
                        <IMPLEMENTATION-DATA-TYPE-REF DEST="IMPLEMENTATION-DATA-
                            TYPE">ImplementationDataTypes/uint8</IMPLEMENTATION-DATA-
                            TYPE-REF>
                        <INVALID-VALUE>
                            <NUMERICAL-VALUE-SPECIFICATION>
                                <VALUE>-1</VALUE>
                            </NUMERICAL-VALUE-SPECIFICATION>
                        </INVALID-VALUE>
                    </SW-DATA-DEF-PROPS-CONDITIONAL>
                </SW-DATA-DEF-PROPS-VARIANTS>
            </SW-DATA-DEF-PROPS>
        </IMPLEMENTATION-DATA-TYPE-ELEMENT>
        <IMPLEMENTATION-DATA-TYPE-ELEMENT>
            <SHORT-NAME>string</SHORT-NAME>
            <CATEGORY>ARRAY</CATEGORY>
            <SUB-ELEMENTS>
                <IMPLEMENTATION-DATA-TYPE-ELEMENT>
                    <SHORT-NAME>character</SHORT-NAME>
                    <CATEGORY>TYPE_REFERENCE</CATEGORY>
                    <ARRAY-SIZE>50</ARRAY-SIZE>
                    <ARRAY-SIZE-SEMANTICS>FIXED-SIZE</ARRAY-SIZE-SEMANTICS>
                    <SW-DATA-DEF-PROPS>
                        <SW-DATA-DEF-PROPS-VARIANTS>
                            <SW-DATA-DEF-PROPS-CONDITIONAL>
                                <IMPLEMENTATION-DATA-TYPE-REF DEST="IMPLEMENTATION-DATA-
                                    TYPE">ImplementationDataTypes/uint8</IMPLEMENTATION-
                                    DATA-TYPE-REF>
                            </SW-DATA-DEF-PROPS-CONDITIONAL>
                        </SW-DATA-DEF-PROPS-VARIANTS>
                    </SW-DATA-DEF-PROPS>
                </IMPLEMENTATION-DATA-TYPE-ELEMENT>
            </SUB-ELEMENTS>
        </IMPLEMENTATION-DATA-TYPE-ELEMENT>
    </SUB-ELEMENTS>
</IMPLEMENTATION-DATA-TYPE>
</ELEMENTS>
</AR-PACKAGE>

```

The interesting part about this definition is the fact that on the implementation level, it was (driven by the definition of the [SwRecordLayout](#)) decided to implement the string as a structure of a size element (that goes by the [shortName](#) “size”) and a value element (that goes by the [shortName](#) “string”) which in turn is defined as an array data type and therefore has a sub-element that goes by the [shortName](#) “character”.

The latter references (in the role [swDataDefProps.implementationDataType](#)) the Platform Data Type “uint8” (that, according to the rules of Platform Data Types, is realized by an [ImplementationDataType](#) “uint8”).

Please note that the [ApplicationPrimitiveDataType](#) named “MyApplication-StringType” references the [SwBaseType](#) named “MyTextBaseType” which is defined in the following XML fragment:

Listing 5.4: Example for the definition of a string [SwBaseType](#)

```
<AR-PACKAGE>
  <SHORT-NAME>BaseTypes</SHORT-NAME>
  <ELEMENTS>
    <SW-BASE-TYPE>
      <SHORT-NAME>MyTextBaseType</SHORT-NAME>
      <BASE-TYPE-SIZE>8</BASE-TYPE-SIZE>
      <BASE-TYPE-ENCODING>UTF-8</BASE-TYPE-ENCODING>
    </SW-BASE-TYPE>
    <SW-BASE-TYPE>
      <SHORT-NAME>uint8BaseType</SHORT-NAME>
      <BASE-TYPE-SIZE>8</BASE-TYPE-SIZE>
    </SW-BASE-TYPE>
  </ELEMENTS>
</AR-PACKAGE>
```

The contribution of this definition of [SwBaseType](#) to the overall definition of a string data type is represented by the definition of the encoding (which is set to UTF-8). However, there is still one important part missing, i.e. the definition of the mapping of [ApplicationPrimitiveDataType](#) to [ImplementationDataType](#) (and vice versa):

Listing 5.5: Example for the definition of the applicable [DataTypeMappingSet](#)

```
<AR-PACKAGE>
  <SHORT-NAME>DataTypeMappingSets</SHORT-NAME>
  <ELEMENTS>
    <DATA-TYPE-MAPPING-SET>
      <SHORT-NAME>theExample</SHORT-NAME>
      <DATA-TYPE-MAPS>
        <DATA-TYPE-MAP>
          <APPLICATION-DATA-TYPE-REF DEST="APPLICATION-PRIMITIVE-DATA-TYPE"
            BASE="default">ApplicationDataTypes/MyApplicationStringType</APPLICATION-DATA-TYPE-REF>
          <IMPLEMENTATION-DATA-TYPE-REF DEST="IMPLEMENTATION-DATA-TYPE"
            BASE="default">ImplementationDataTypes/
            MyImplementationStringType</IMPLEMENTATION-DATA-TYPE-REF>
        </DATA-TYPE-MAP>
      </DATA-TYPE-MAPS>
    </DATA-TYPE-MAPPING-SET>
  </ELEMENTS>
```

</AR-PACKAGE>

As mentioned before, the definition of an `ImplementationDataType` that corresponds to an `ApplicationPrimitiveDataType` of category `STRING` can be to some extent derived from the `ApplicationPrimitiveDataType.swDataDefProps.swRecordLayout`.

[TPS_SWCT_01570] `DataTypeMap` is mandatory in the presence of `ApplicationPrimitiveDataType.swDataDefProps.swRecordLayout` [The definition of a `DataTypeMap` is mandatory even if an `ImplementationDataType` has been derived from an `ApplicationPrimitiveDataType` that defines a `SwRecordLayout`.]()

One motivation for the existence of [TPS_SWCT_01570] is that the integrator of an AUTOSAR ECU may rightfully decide to take a different `ImplementationDataType` other than the one that has been generated on the basis of the `SwRecordLayout`.

5.2.4.2 Application Composite Data Types

[TPS_SWCT_01247] `ApplicationArrayDataType` and `ApplicationRecordDataType` [The meta-classes `ApplicationArrayDataType` and `ApplicationRecordDataType` (details are depicted in Figure 5.9) provide the means to define composite data types.

Such a composite data type is required if the application software wants to have access to the individual elements of the composite as well as to do operations with the whole composite, e.g. wants to communicate the complete record or array in a single transaction.

It is possible to use a combination of `ApplicationArrayDataType` and `ApplicationRecordDataType`, so that an `ApplicationArrayDataType` could be defined as `ApplicationRecordElement` of a `ApplicationRecordDataType` and in the same manner a `ApplicationRecordDataType` could be used as the base type of an `ApplicationArrayDataType`. The creation of nested `ApplicationCompositeDataTypes` is also possible.]()

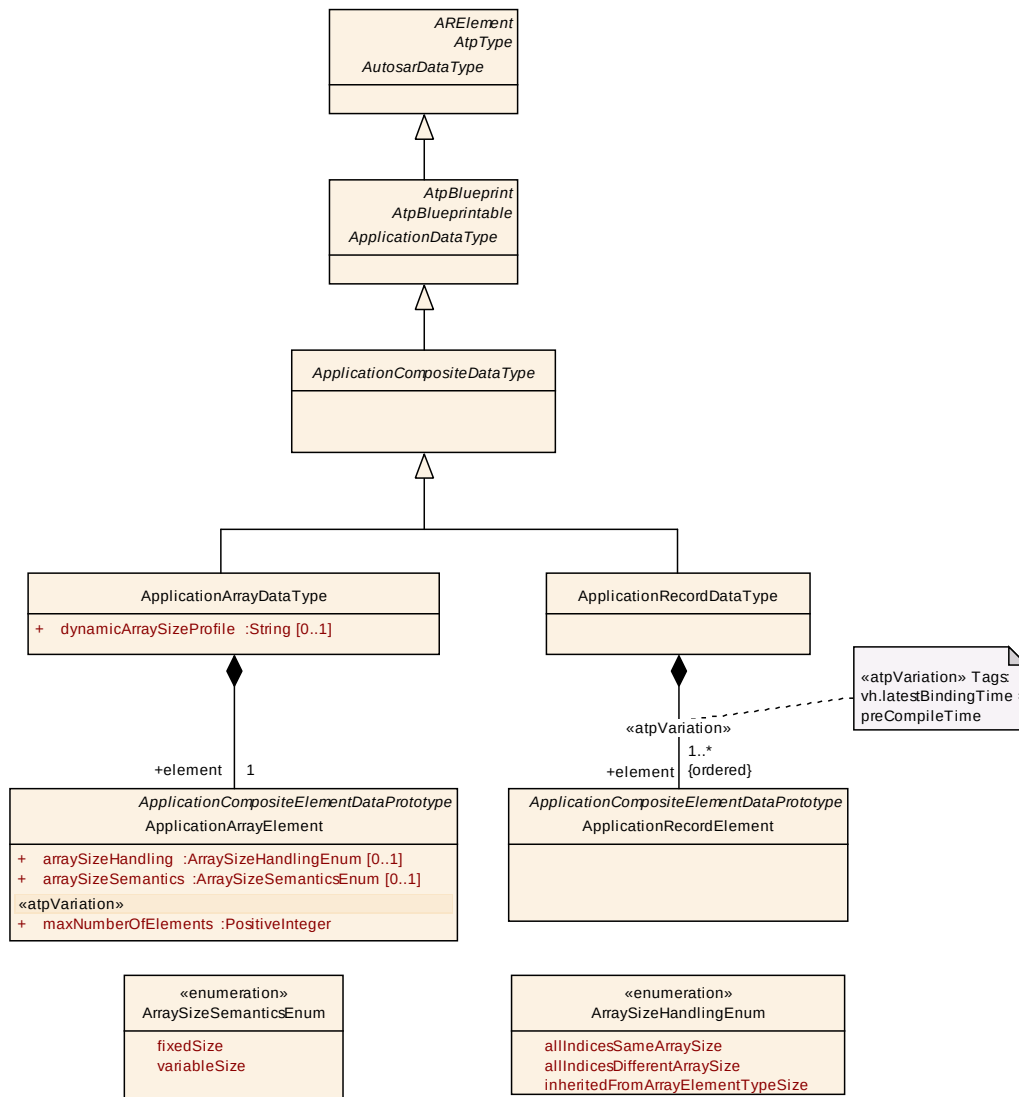


Figure 5.9: Summary of `ApplicationCompositeDataType`

5.2.4.2.1 ApplicationArrayDataType

[TPS_SWCT_01078] Configurable array size | An `ApplicationArrayDataType` may⁴ contain `maxNumberOfElements` `ApplicationArrayElements`.

Each of these `ApplicationArrayElements` has the same data type.

When referring to an element of an `ApplicationArrayDataType` within a software-component description, the element-index runs from 0 to the value of `maxNumberOfElements-1`. | ([RS_SWCT_03144](#))

⁴This applies although the multiplicity in the meta-model is 1. In fact, it would be possible to model `ApplicationArrayDataType` without `ApplicationArrayElement`. The latter exists only so that it can be the target of a reference within an AUTOSAR XML file

Class	ApplicationArrayType			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::Datatypes			
Note	An application data type which is an array, each element is of the same application data type. Tags: atp.recommendedPackage=ApplicationDataTypes			
Base	AElement, ARObjct, ApplicationCompositeDataType , ApplicationDataType , AtpBlueprint , AtpBlueprintable , AtpClassifier , AtpType , AutosarDataType , CollectableElement , Identifiable , MultilanguageReferrable , PackageableElement , Referrable			
Attribute	Datatype	Mul.	Kind	Note
dynamicArraySizeProfile	String	0..1	attr	Specifies the profile which the array will follow if it is a variable size array.
element	ApplicationArrayElement	1	aggr	This association implements the concept of an array element. That is, in some cases it is necessary to be able to identify single array elements, e.g. as input values for an interpolation routine.

Table 5.12: ApplicationArrayType

Class	ApplicationArrayElement			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::DataPrototypes			
Note	Describes the properties of the elements of an application array data type.			
Base	ARObject, ApplicationCompositeElementDataPrototype , AtpFeature , AtpPrototype , DataPrototype , Identifiable , MultilanguageReferrable , Referrable			
Attribute	Datatype	Mul.	Kind	Note
arraySizeHandling	ArraySizeHandlingEnum	0..1	attr	The way how the size of the array is handled.
arraySizeSemantics	ArraySizeSemanticsEnum	0..1	attr	This attribute controls how the information about the array size shall be interpreted.
maxNumberOfElements	PositiveInteger	1	attr	The maximum number of elements that the array can contain. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime

Table 5.13: ApplicationArrayElement

Please note that the information about the number of elements of a specific [ApplicationArrayType](#) is not absolute but allows for further interpretation.

[TPS_SWCT_01076] Number of elements of a specific [ApplicationArrayDataType](#) might vary at run-time [That is, there are cases where the number of elements of a specific [ApplicationArrayType](#) might vary at run-time.

To be precise, the number of elements might vary between 0 and the value denoted by [maxNumberOfElements](#).

For this purpose an additional attribute [arraySizeSemantics](#) is available that can be used to clarify the meaning of [maxNumberOfElements](#).

For clarification, it might indeed happen that the actual number of elements in a specific `ApplicationArrayDataType` yields 0 simply because the respective `DataPrototype` is part of a higher-level protocol where under certain circumstances the `DataPrototype` of `ApplicationArrayDataType` is simply not required for expressing a given semantics. `⌋(RS_SWCT_03180, RS_SWCT_03181, RS_SWCT_03144)`

Enumeration	ArraySizeSemanticsEnum
Package	M2::AUTOSARTemplates::CommonStructure::ImplementationDataTypes
Note	This type controls how the information about the number of elements in an <code>ApplicationArrayDataType</code> is to be interpreted.
Literal	Description
fixedSize	This means that the <code>ApplicationArrayDataType</code> will always have a fixed number of elements.
variableSize	This implies that the actual number of elements in the <code>ApplicationArrayDataType</code> might vary at run-time. The value of <code>arraySize</code> represents the maximum number of elements in the array.

Table 5.14: ArraySizeSemanticsEnum

Please note that the ability to define the semantic meaning of `maxNumberOfElements` is not only limited to the application data type level. The same approach also applies for `ImplementationDataType`.

[constr_1152] category of `ApplicationArrayElement` and `AutosarDataType` referenced in the role `type` shall be kept in sync `⌈` The value of `category` of an `ApplicationArrayElement` shall always be identical to the value of `category` of the `AutosarDataType` referenced by the `ApplicationArrayElement`. `⌋()`

[TPS_SWCT_01601] Size Indicator shall be updated by software-component `⌈` If a software-component changes the number of valid elements in a variable size array, it shall also update the `Size Indicator` in the `ImplementationDataType`. `⌋(RS_SWCT_03181)`

[TPS_SWCT_01602] Size Indicator shall be read by the software-component `⌈` If a software-component receives a variable size array, it shall use the `Size Indicator` in the `ImplementationDataType` to determine the number of valid elements in the array. `⌋(RS_SWCT_03181)`

Enumeration	ArraySizeHandlingEnum
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::Datatypes
Note	This enumeration defines different ways to handle the sizes of variable size arrays.
Literal	Description
allIndices DifferentArray Size	All elements of the variable size array may have different sizes.
allIndices SameArray Size	All elements of the variable size array have the same size.

inheritedFrom ArrayElement TypeSize	The size of all dimensions of the variable size array is determined by the size of the contained array element.

Table 5.15: ArraySizeHandlingEnum

[TPS_SWCT_01604] Enable `Size Indicator` [To enable the RTE's ability to consider the number of valid elements inside a `Variable-Size Array Data Type` the `ApplicationArrayDataType.dynamicArraySizeProfile` of `ApplicationArrayDataType` and `ApplicationArrayElement.arraySizeHandling` shall be set.]([RS_SWCT_03181](#))

[TPS_SWCT_01605] Semantics of `ApplicationArrayElement.arraySizeHandling` [The attribute `ApplicationArrayElement.arraySizeHandling` specifies how the size is determined in case of multi-dimensional variable size array.]([RS_SWCT_03181](#))

This allows to specify coherencies between the sizes of the nested variable size arrays in case of multiple dimensions.

With a suitable `ImplementationDataType`, it is possible to enable other software-components, RTE, and other BSW modules to make use of the `Size Indicator` and only transfer the valid data elements from the sender to the receiver.

[TPS_SWCT_01606] Internal structure of mapped `ImplementationDataType` [The attribute `dynamicArraySizeProfile` specifies which internal structure the `ImplementationDataType` that is mapped to the `ApplicationDataType` shall follow.]([RS_SWCT_03181](#))

[TPS_SWCT_01607] Profiles for internal structure of mapped `ImplementationDataType` [For the structure of the `ImplementationDataType` that is mapped to the `ApplicationDataType` the following profiles are defined for `dynamicArraySizeProfile`: `VSA_LINEAR`, `VSA_SQUARE`, `VSA_RECTANGULAR`, and `VSA_FULLY_FLEXIBLE`.]([RS_SWCT_03181](#))

[TPS_SWCT_01608] Custom profiles for internal structure of mapped `ImplementationDataType` [Custom profiles can be added to `dynamicArraySizeProfile`. They shall have a company-specific prefix.]([RS_SWCT_03181](#))

As it is a general rule for the definition of custom profiles or values of `category`, the custom value should start with a company-specific prefix in order to avoid clashes with later extensions of the AUTOSAR standard.

`dynamicArraySizeProfile` is used to specify how the number of elements of the multiple dimensions of a variable size array correlate. They could be totally independent (`VSA_FULLY_FLEXIBLE`) on the one hand or each dimension has the same number of valid elements (`VSA_SQUARE`).

[TPS_SWCT_01623] Justification for the existence of attributes `ApplicationArrayDataType.dynamicArraySizeProfile` and `ApplicationArrayEle-`

ment.arraySizeHandling [At the first glance, the two attributes `ApplicationArrayDataType.dynamicArraySizeProfile` and `ApplicationArrayElement.arraySizeHandling` seem equivalent.

However, both are needed because they have to be used if multi dimensional variable size arrays have to be described. In this case, multiple combinations of sizes could occur which cannot be specified beforehand.]([RS_SWCT_03181](#))

The `ImplementationDataType` has to follow certain rules depending on the chosen profile. See chapter [5.2.5](#) for details.

[constr_1314] Profile `VSA_LINEAR` for `ApplicationArrayDataType` [If the `dynamicArraySizeProfile` of `ApplicationArrayDataType` is set to `VSA_LINEAR`, the contained `ApplicationArrayElement` shall fulfill **all** of the following conditions:

- The attribute `ApplicationArrayElement.arraySizeSemantics` shall set to the value `variableSize`.
- The attribute `ApplicationArrayElement.maxNumberOfElements` shall be defined.
- The attribute `ApplicationArrayElement.arraySizeHandling` shall be set to the value `allIndicesSameArraySize`.
- The `ApplicationArrayElement` shall be typed by an `ApplicationDataType` that is not an `ApplicationArrayDataType` where the attribute `dynamicArraySizeProfile` exists.

]()

The part of **[constr_1314]** that demands that *the `ApplicationArrayElement` shall be typed by an `ApplicationDataType` that is not an `ApplicationArrayDataType` where the attribute `dynamicArraySizeProfile` exists* basically boils down to the simple explanation that the “leaf” data type of the `Variable-Size Array Data Type` can be anything but a `Variable-Size Array Data Type`.

[constr_1315] Profile `VSA_SQUARE` for `ApplicationArrayDataType` [If the `dynamicArraySizeProfile` of `ApplicationArrayDataType` is set to `VSA_SQUARE`, the contained `ApplicationArrayElement` shall fulfill **all** of the following conditions:

- The attribute `ApplicationArrayElement.arraySizeSemantics` shall be set to the value `variableSize`.
- The attribute `ApplicationArrayElement.maxNumberOfElements` shall not be defined.
- The attribute `ApplicationArrayElement.arraySizeHandling` shall be set to the value `inheritedFromArrayElementTypeSize`.

- The `ApplicationArrayElement` shall be typed by an `ApplicationArrayDataType`.

The referred `ApplicationArrayDataType` shall refer over a chain (under consideration of the number of dimensions of the “root” `ApplicationArrayDataType`) of nested `ApplicationArrayDataTypes` with `ApplicationArrayElements` to an `ApplicationDataType` that is **not** an `ApplicationArrayDataType` where the attribute `dynamicArraySizeProfile` exists.

The last `ApplicationArrayDataType` in that chain shall have an `ApplicationArrayElement` that fulfills **all** of the following conditions:

- The attribute `ApplicationArrayElement.arraySizeSemantics` shall be set to the value `variableSize`.
- The attribute `ApplicationArrayElement.maxNumberOfElements` shall be defined.
- The attribute `ApplicationArrayElement.arraySizeHandling` set to the value `allIndicesSameArraySize`.

All `ApplicationArrayDataTypes` before shall have an `ApplicationArrayElement` that fulfills **all** of the following conditions:

- The attribute `ApplicationArrayElement.arraySizeSemantics` shall be set to the value `variableSize`.
- The attribute `ApplicationArrayElement.maxNumberOfElements` shall not be defined.
- The attribute `ApplicationArrayElement.arraySizeHandling` shall be set to the value `inheritedFromArrayElementTypeSize`.
- The `ApplicationArrayElement` shall be typed by an `ApplicationArrayDataType`.

]()

The part of [constr_1315], [constr_1316], and [constr_1317] that demands that *the referred `ApplicationArrayDataType` shall refer over a chain (under consideration of the number of dimensions of the “root” `ApplicationArrayDataType`) of nested `ApplicationArrayDataTypes` with `ApplicationArrayElements` to an `ApplicationDataType` that is **not** an `ApplicationArrayDataType` where the attribute `dynamicArraySizeProfile` exists* basically boils down to the simple explanation that the “leaf” data type of the `Variable-Size Array Data Type` can be anything but a `Variable-Size Array Data Type`.

[constr_1316] Profile `VSA_RECTANGULAR` for `ApplicationArrayDataType` [If the `dynamicArraySizeProfile` of `ApplicationArrayDataType` is set to `VSA_RECTANGULAR` the contained `ApplicationArrayElement` shall fulfill **all** of the following conditions:

- The attribute `ApplicationArrayElement.arraySizeSemantics` shall be set to the value `variableSize`.
- The attribute `ApplicationArrayElement.maxNumberOfElements` shall be defined.
- The attribute `ApplicationArrayElement.arraySizeHandling` shall be set to the value `allIndicesSameArraySize`.
- The `ApplicationArrayElement` shall be typed by an `ApplicationArrayDataType`.

The referred `ApplicationArrayDataType` shall refer over a chain (under consideration of the number of dimensions of the “root” `ApplicationArrayDataType`) of nested `ApplicationArrayDataTypes` with `ApplicationArrayElements` to an `ApplicationDataType` that is **not** an `ApplicationArrayDataType` where the attribute `dynamicArraySizeProfile` exists.

The last `ApplicationArrayDataType` in that chain shall have an `ApplicationArrayElement` that fulfills **all** of the following conditions:

- The attribute `ApplicationArrayElement.arraySizeSemantics` shall be set to the value `variableSize`.
- The attribute `ApplicationArrayElement.maxNumberOfElements` shall be defined.
- The attribute `ApplicationArrayElement.arraySizeHandling` shall be set to the value `allIndicesSameArraySize`.

All `ApplicationArrayDataTypes` before shall have an `ApplicationArrayElement` that fulfills **all** of the following conditions:

- The attribute `ApplicationArrayElement.arraySizeSemantics` shall set to the value `variableSize`
- The attribute `ApplicationArrayElement.maxNumberOfElements` shall be defined.
- The attribute `ApplicationArrayElement.arraySizeHandling` shall be set to the value `allIndicesSameArraySize`.
- The `ApplicationArrayElement` shall be typed by an `ApplicationArrayDataType`.

]()

[constr_1317] Profile `VSA_FULLY_FLEXIBLE` for `ApplicationArrayDataType`

[If the `dynamicArraySizeProfile` of `ApplicationArrayDataType` is set to `VSA_FULLY_FLEXIBLE`, the contained `ApplicationArrayElement` shall fulfill **all** of the following conditions:

- The attribute `ApplicationArrayElement.arraySizeSemantics` shall be set to the value `variableSize`.
- The attribute `ApplicationArrayElement.maxNumberOfElements` shall be defined.
- The attribute `ApplicationArrayElement.arraySizeHandling` shall be set to the value `allIndicesDifferentArraySize`.
- The `ApplicationArrayElement` shall be typed by an `ApplicationArrayDataType`.

The referred `ApplicationArrayDataType` shall refer over a chain (under consideration of the number of dimensions of the “root” `ApplicationArrayDataType`) of nested `ApplicationArrayDataTypes` with `ApplicationArrayElements` to an `ApplicationDataType` that is **not** an `ApplicationArrayDataType` where the attribute `dynamicArraySizeProfile` exist.

The last `ApplicationArrayDataType` in that chain shall have an `ApplicationArrayElement` that fulfills **all** of the following conditions:

- The attribute `ApplicationArrayElement.arraySizeSemantics` shall be set to the value `variableSize`.
- The attribute `ApplicationArrayElement.maxNumberOfElements` shall be defined.
- The attribute `ApplicationArrayElement.arraySizeHandling` shall be set to the value `allIndicesSameArraySize`.

All `ApplicationArrayDataTypes` before shall have an `ApplicationArrayElement` that fulfills **all** of the following conditions:

- The attribute `ApplicationArrayElement.arraySizeSemantics` shall be set to the value `variableSize`.
- The attribute `ApplicationArrayElement.maxNumberOfElements` shall be defined.
- The attribute `ApplicationArrayElement.arraySizeHandling` shall be set to the value `allIndicesDifferentArraySize`.
- The `ApplicationArrayElement` shall be typed by an `ApplicationArrayDataType`.

⌋()

For examples see Appendix [E.1](#).

[TPS_SWCT_01256] Definition of multi-dimensional array data types [In order to describe multi dimensional arrays an `ApplicationArrayElement` references again another `ApplicationArrayDataType`. Hereby, one `ApplicationArrayDataType` per dimension is required.

This multiple dimensions do have a well-defined correlation to the individual dimensions of an `ImplementationDataType` of category `ARRAY` when the `ApplicationArrayDataType` is mapped to an `ImplementationDataType` as described in section 5.2.2

The `ApplicationArrayElements` are mapping in the order of the `ApplicationArrayElement` to `ApplicationArrayDataType` references to `ImplementationDataTypeElements` in the order of first `ImplementationDataTypeElement` of the `ImplementationDataType` to leaf `ImplementationDataTypeElement`.

In other words the `ApplicationArrayElement` of the top level `ApplicationArrayDataType` relates to the first `ImplementationDataTypeElement` of the `ImplementationDataType`. The `ApplicationArrayElement` of the referenced `ApplicationArrayDataTypes` relates to the sub `ImplementationDataTypeElements` in the order of the `ApplicationArrayElement` -> `ApplicationArrayDataType` references. |(RS_SWCT_03216)

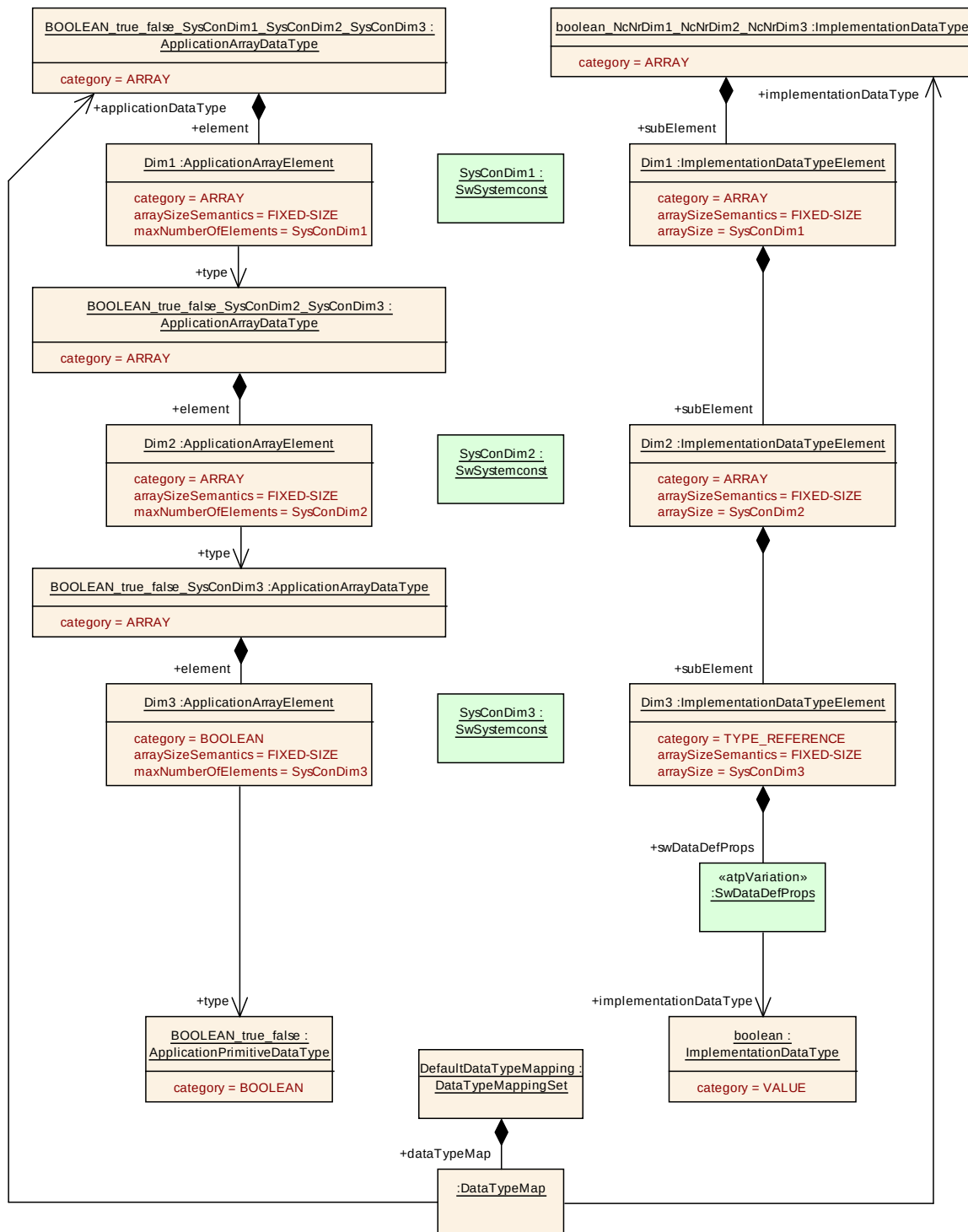


Figure 5.10: Example of a three dimensional array type

Figure 5.10 shows a three dimensional array described with a set of [ApplicationArrayDataTypes](#) on the left hand side. The array element is typed by an [ApplicationPrimitiveDataType](#) of category [BOOLEAN](#). On the right hand side the implementation of the three dimensional array is described with an [ImplementationDataType](#) which contains three nested [ImplementationDataTypeElements](#).

Matching `ApplicationArrayElements` and `ImplementationDataTypeElements` are shown on the same layer. For the sake of clarity correlating `maxNumberOfElements` and `arraySize` attributes are described with the identical instance of a `SwSystemconst` instead of a value. Further details of variant rich M1 models are not in the scope of this example.

The data type of the array element is described by the `ApplicationArrayDataType` with the means of a `ApplicationPrimitiveDataType` of category `BOOLEAN`. In order to fulfill `[constr_1152]` the category of `ApplicationArrayElement` “Dim3” is set to `BOOLEAN`. This `ApplicationPrimitiveDataType` “`BOOLEAN`” correlates to the `ImplementationDataType` “`boolean`” of category `VALUE` which is typically the boolean type of the AUTOSAR Platform Types. Please note here `[constr_1063]`.

5.2.4.2.2 ApplicationRecordDataType

[TPS_SWCT_01249] `ApplicationRecordDataType` [A declaration of `ApplicationRecordDataType` describes a non-empty set of objects, each of which has a unique identifier with respect to the `ApplicationRecordDataType` and each has an own `ApplicationDataType`. The `shortName` of each `ApplicationRecordElement` within the scope of an `ApplicationRecordDataType` shall be unique.] (`RS_SWCT_03216`)

Class	ApplicationRecordDataType			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::Datatypes			
Note	An application data type which can be decomposed into prototypes of other application data types. Tags: atp.recommendedPackage=ApplicationDataTypes			
Base	<code>ARElement</code> , <code>ARObject</code> , <code>ApplicationCompositeDataType</code> , <code>ApplicationDataType</code> , <code>AtpBlueprint</code> , <code>AtpBlueprintable</code> , <code>AtpClassifier</code> , <code>AtpType</code> , <code>AutosarDataType</code> , <code>CollectableElement</code> , <code>Identifiable</code> , <code>MultilanguageReferrable</code> , <code>PackageableElement</code> , <code>Referrable</code>			
Attribute	Datatype	Mul.	Kind	Note
element (ordered)	<code>ApplicationRecordElement</code>	1..*	aggr	Specifies an element of a record. The aggregation of <code>ApplicationRecordElement</code> is subject to variability with the purpose to support the conditional existence of elements inside a <code>ApplicationrecordDataType</code> . Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime

Table 5.16: ApplicationRecordDataType

Class	ApplicationRecordElement			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::DataPrototypes			
Note	Describes the properties of one particular element of an application record data type.			
Base	ARObject, ApplicationCompositeElementDataPrototype , AtpFeature, AtpPrototype, DataPrototype , Identifiable , MultilanguageReferrable, Referrable			
Attribute	Datatype	Mul.	Kind	Note
—	—	—	—	—

Table 5.17: ApplicationRecordElement

5.2.5 Implementation Data Type

[TPS_SWCT_01250] [ImplementationDataType](#) has been introduced to optimize the formal support for data type handling on the implementation level [The concept of an [ImplementationDataType](#) has been introduced to optimize the formal support for data type handling on the implementation level.

That is, an [ImplementationDataType](#) conceptually corresponds to the level of (C) source code. For example, [ImplementationDataTypes](#) have a direct impact on the contract (please find an explanation of this term in [2]) of a software-component and the RTE.]([RS_SWCT_03217](#))

	Root Element				Attribute Existence per Category						
	ImplementationDataType	ImplementationDataTypeElement	SwPointerTargetProps	SwServiceArg	VALUE	DATA_REFERENCE	FUNCTION_REFERENCE	TYPE_REFERENCE	STRUCTURE	UNION	ARRAY
Attributes of SwDataDefProps											
additionalNativeTypeQualifier	x	x	x	x	0..1	0..1	0..1	0..1	0..1	0..1	0..1
annotation	x	x	x	x	*	*	*	*	*	*	*
baseType	x	x	x	x	1						
compuMethod	x	x	x	x	0..1			0..1			
dataConstr	x	x	x	x	0..1			0..1			
displayFormat	x	x			0..1				0..1	0..1	0..1
implementationDataType	x	x	x	x				1			
invalidValue	x	x	x		0..1			0..1	0..1 ⁵		0..1 ⁶
stepSize	x	x			0..1						
swAddrMethod	x	x	x		0..1	0..1	0..1	0..1	0..1	0..1	0..1

⁵There is a use case for the definition of an [invalidValue](#) for category [ARRAY](#) and therefore category [STRUCTURE](#) is also supported for the sake of symmetry.

⁶This represents an exception such that it would make sense to use an entire [ArrayValueSpecification](#) as the [invalidValue](#) because a string semantically is more than just a bunch of characters in a row.

	Root Element				Attribute Existence per Category						
	ImplementationDataType	ImplementationDataTypeElement	SwPointerTargetProps	SwServiceArg	VALUE	DATA_REFERENCE	FUNCTION_REFERENCE	TYPE_REFERENCE	STRUCTURE	UNION	ARRAY
Attributes of SwDataDefProps											
swAlignment	x				0..1	0..1	0..1		0..1	0..1	0..1
swBitRepresentation											
swCalibrationAccess	x	x			0..1				0..1	0..1	0..1
swCalprmAxisSet											
swComparisonVariable											
swDataDependency											
swHostVariable											
swImplPolicy	x		x	x	0..1	0..1	0..1	0..1	0..1	0..1	0..1
swIntendedResolution											
swInterpolationMethod											
swIsVirtual											
swPointerTargetProps	x	x	x	x		1	1				
swPointerTargetProps .swDataDefProps	x	x	x	x		1					
swPointerTargetProps .functionPointerSignature	x	x	x	x			1				
swRecordLayout											
swRefreshTiming	x	x	x	x	0..1				0..1	0..1	0..1
swTextProps											
swValueBlockSize											
unit											
valueAxisDataType											
Other Attributes											
subElement: Implementation- DataTypeElement	x	x							1..*	1..*	1
subElement .arraySizeSemantics	x	x									0..1
subElement.arraySize	x	x									1

Table 5.18: Allowed Attributes vs. **category** for **ImplementationDataType**

[TPS_SWCT_01251] Limited set of values for **category** are applicable for **ImplementationDataType** [Like any **AutosarDataType**, also the data types on implementation level are characterized by its **category** and its **SwDataDefProps**. For a given **category**, only a limited set of attributes of the **SwDataDefProps** makes sense.](*RS_SWCT_03217*)

[constr_1009] **SwDataDefProps** applicable to **ImplementationDataTypes** [A complete list of the **SwDataDefProps** and other attributes and their multiplicities which are allowed for a given **category** is shown in table 5.18.]()

This list makes use of the `SwDataDefProps` and other meta-model elements which are explained in detail in the further sections of this chapter.

[constr_1158] Applicable `category`s for attribute `ImplementationDataType.swDataDefProps.compuMethod` [The definition of the reference `ImplementationDataType.swDataDefProps.compuMethod` is restricted to a `CompuMethod` of either `category` `BITFIELD_TEXTTABLE` or `category` `TEXTTABLE` (these might be seen as implementation specific in certain cases).]()

[constr_1383] Existence of `CompuMethod` and `DataConstr` for `ImplementationDataTypes` of `category` `TYPE_REFERENCE` [The existence of `ImplementationDataType.swDataDefProps.compuMethod` and `ImplementationDataType.swDataDefProps.dataConstr` for `ImplementationDataTypes` of `category` `TYPE_REFERENCE` is only allowed if the respective `ImplementationDataType`, after all type references are resolved, ends up in an `ImplementationDataType` of `category` `VALUE`.]()

Please note that, as a consequence of the existence of [constr_1383], it is possible that the elements of a composite `ImplementationDataType` define individual `CompuMethods`. However, the definition of **one** `CompuMethod` that applies to the **entire** composite `ImplementationDataType` is not supported.

[TPS_SWCT_01252] `ImplementationDataType` can express concepts not available on application level [As a consequence of the specific focus, it is possible to express concepts with an `ImplementationDataType` that are not supported on the application level, i.e. by `ApplicationDataType`:

- `ImplementationDataType` supports the definition of pointers
- It is possible to define “alias” names just as in a `typedef`
- It is possible to define nested `ImplementationDataTypes` but in contrast to the concept implemented for `ApplicationDataType` these implement a direct aggregation of sub-elements rather than applying the type-prototype pattern.

] ([RS_SWCT_03217](#))

The general structure of `ImplementationDataType` is sketched in Figure 5.11. If a specific `ImplementationDataType` is supposed to define a composite data type the `ImplementationDataType` aggregates `ImplementationDataTypeElements`.

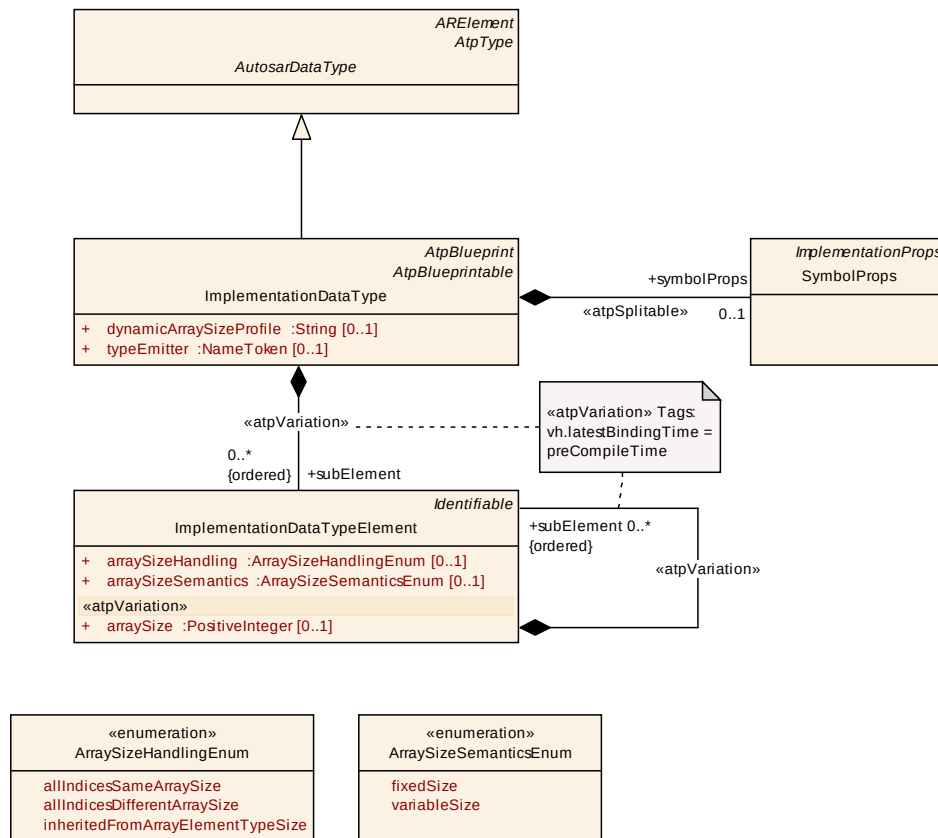


Figure 5.11: **ImplementationDataType** overview

Class	ImplementationDataType			
Package	M2::AUTOSARTemplates::CommonStructure::ImplementationDataTypes			
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	AElement, AObject, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, Autosar DataType, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Attribute	Datatype	Mul.	Kind	Note
dynamicArraySizeProfile	String	0..1	attr	Specifies the profile which the array will follow in case this data type is a variable size array.
subElement (ordered)	ImplementationDataTypeElement	*	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

Attribute	Datatype	Mul.	Kind	Note
symbolProps	SymbolProps	0..1	aggr	This represents the SymbolProps for the ImplementationDataType. Stereotypes: atpSplittable Tags: atp.Splitkey=shortName
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 5.19: ImplementationDataType

[TPS_SWCT_01253] Rules apply for the usage of the attribute **ImplementationDataType.typeEmitter** [The following set of rules applies for the usage of the attribute **ImplementationDataType.typeEmitter**:

- If the value of attribute **typeEmitter** is NOT defined **and** a **nativeDeclaration** is provided the RTE generator shall generate the corresponding data type definition⁷.
- If the value of attribute **typeEmitter** is set to “RTE” **and** a **nativeDeclaration** is provided the RTE generator shall generate the corresponding data type definition.
- If the value of the attribute **typeEmitter** is set to “RTE” **and** no **nativeDeclaration** is provided the RTE generator shall issue an error message.
- If the value of attribute **typeEmitter** is set to anything else but “RTE” the RTE generator shall silently **not** generate the corresponding data type definition regardless of the existence of **nativeDeclaration** attribute.

](RS_SWCT_03217)

Note that the rules listed above imply that the allowed values of the attribute **typeEmitter** are not constrained with the singular exception that the definition of the behavior in case of “RTE” is claimed by AUTOSAR. Other values can be provided; the consequences of this provision are implementation-dependent and outside the scope of the definition of the AUTOSAR standard.

[TPS_SWCT_01248] Nested definition of **ImplementationDataType** [If an **ImplementationDataTypeElement** also represents a composite data type it can aggregate **ImplementationDataTypeElements** in the role of **subElement**. Again, the type-prototype pattern does not apply in this case.](RS_SWCT_03217)

[constr_1106] Structure shall have at least one element [An **ImplementationDataType** or **ImplementationDataTypeElement** of category **STRUCTURE** shall own at least one **ImplementationDataTypeElement**.]()

⁷This rule represents the behavior before the attribute **typeEmitter** was introduced. The rule has specifically been added in order to support a backwards-compatible behavior.

[constr_1107] Union shall have at least one element [An [ImplementationDataType](#) or [ImplementationDataTypeElement](#) of category UNION shall own at least one [ImplementationDataTypeElement](#).]()

Class	ImplementationDataTypeElement			
Package	M2::AUTOSARTemplates::CommonStructure::ImplementationDataTypes			
Note	Declares a data object which is locally aggregated. Such an element can only be used within the scope where it is aggregated. This element either consists of further subElements or it is further defined via its swDataDefProps. There are several use cases within the system of ImplementationDataTypes for such a local declaration: • It can represent the elements of an array, defining the element type and array size • It can represent an element of a struct, defining its type • It can be the local declaration of a debug element.			
Base	ARObject, Identifiable , MultilanguageReferrable, Referrable			
Attribute	Datatype	Mul.	Kind	Note
arraySize	PositiveInteger	0..1	attr	The existence of this attributes (if bigger than 0) defines the size of an array and declares that this ImplementationDataTypeElement represents the type of each single array element. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime
arraySizeHandling	ArraySizeHandlingEnum	0..1	attr	The way how the size of the array is handled in case of a variable size array.
arraySizeSemantics	ArraySizeSemanticsEnum	0..1	attr	This attribute controls the meaning of the value of the array size.
subElement	ImplementationDataTypeElement	*	aggr	Element of an array, struct, or union in case of a nested declaration (i.e. without using "typedefs"). 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
swDataDefProps	SwDataDefProps	0..1	aggr	The properties of this ImplementationDataTypeElement.

Table 5.20: ImplementationDataTypeElement

[TPS_SWCT_01254] ImplementationDataType with array semantics [Of course, it is also possible to define an [ImplementationDataType](#) that provides array semantics.]([RS_SWCT_03217](#))

[TPS_SWCT_01006] `ImplementationDataType.subElement.arraySize` shall be used to define the size of the array [The primitive attribute `ImplementationDataType.subElement.arraySize` shall be used to define the size of the array.]()

[TPS_SWCT_01007] Semantics of array index [For an `ImplementationDataType` that implements an array data type, the semantics of the array index is such that

- it shall start with the value 0
- it shall run to the value of `arraySize - 1`

]()

[constr_1105] Value of `arraySize` [The value of the attribute `arraySize` of an `ImplementationDataTypeElement` owned by an `ImplementationDataType` or `ImplementationDataTypeElement` of category ARRAY shall be greater than 0.]()

[TPS_SWCT_01478] Array size is defined as an attribute of the `ImplementationDataTypeElement` [Please note that the array size is **not** defined as an attribute of the `ImplementationDataType` which stands for the whole array. It is actually defined as an attribute of the `ImplementationDataTypeElement` which is describing the array element (note that the same pattern is used in `ApplicationArrayDataType`).]()

Consequently, if a “struct” element represents an array this specific struct-element is given by an `ImplementationDataTypeElement` of category ARRAY which in turn aggregates another `ImplementationDataTypeElement` of e.g. category VALUE representing the array element and containing the size.

[TPS_SWCT_01255] Indicate whether the array is supposed to have a fixed size or whether the actual size might change during run-time [It is also possible to indicate whether the array is supposed to have a fixed size or whether the actual size might change during run-time.]([RS_SWCT_03217](#))

In the same way as for `ApplicationDataTypes`, it is also possible to specify a `Size Indicator` of a variable size array which holds the number of valid elements of the array in the `ImplementationDataType`.

Please find more information about this topic in section [5.2.4.2](#).

[TPS_SWCT_01622] Modeling of a `Variable-Size Array Data Type` only with `ImplementationDataType` [The modeling of a `Variable-Size Array Data Type` does not require the existence of an `ApplicationCompositeDataType` and a `DataTypeMap`. A `Variable-Size Array Data Type` can be created by just setting up an `ImplementationDataType`.]([RS_SWCT_03181](#))

[TPS_SWCT_01610] Modeling of a `Variable-Size Array Data Type` with `Size Indicator` enabled [An `ImplementationDataType` with category STRUCTURE where the attribute `ImplementationDataType.dynamicArray-`

`SizeProfile` exists represents a `Variable-Size Array Data Type` with `Size Indicator` enabled.

For the sake of a proper definition of terminology, this `ImplementationDataType` shall be called the `VSA ImplementationDataType`. [|\(RS_SWCT_03181\)](#)

[TPS_SWCT_01650] Structure of the `VSA ImplementationDataType` [|](#) The `VSA ImplementationDataType` shall consist of

- an `ImplementationDataTypeElement` representing the `Size Indicator` and
- an `ImplementationDataTypeElement` representing the Payload of the `Variable-Size Array Data Type` (see section [2.8.1.2](#)).

For the sake of a proper definition of terminology, these `ImplementationDataTypeElements` shall be called the `VSA Size Indicator ImplementationDataTypeElement` and the `VSA Payload ImplementationDataTypeElement` respectively. [|\(RS_SWCT_03181\)](#)

[TPS_SWCT_01612] `arraySizeHandling` specifies how the size is determined [|](#) `arraySizeHandling` specifies how the size is determined in case of multi-dimensional variable size array. [|\(RS_SWCT_03181\)](#)

The statement made by [\[TPS_SWCT_01612\]](#) allows the specification of coherencies between the sizes of the nested variable size arrays in case of multiple dimensions.

[TPS_SWCT_01613] Internal structure of mapped `ImplementationDataType` [|](#) The attribute `dynamicArraySizeProfile` specifies which internal structure the `ImplementationDataType` shall follow. [|\(RS_SWCT_03181\)](#)

[TPS_SWCT_01614] Profiles for internal structure of mapped `ImplementationDataType` [|](#) For the structure of the `ImplementationDataType` the following profiles are defined for `dynamicArraySizeProfile`: `VSA_LINEAR`, `VSA_SQUARE`, `VSA_RECTANGULAR` and `VSA_FULLY_FLEXIBLE`. [|\(RS_SWCT_03181\)](#)

[TPS_SWCT_01615] Custom profiles for internal structure of mapped `ImplementationDataType` [|](#) Custom profiles can be added to `dynamicArraySizeProfile`. They shall have a company-specific prefix. [|\(RS_SWCT_03181\)](#)

For reasons of readability and understandability the following constraints focus on the payload of the `Variable-Size Array Data Type` only. For the `Size Indicator` additional individual constraints do apply.

[constr_1318] Profile `VSA_LINEAR` for `ImplementationDataType` [|](#) If the value of attribute `ImplementationDataType.dynamicArraySizeProfile` is set to `VSA_LINEAR`, the `ImplementationDataType` shall aggregate a `VSA Payload ImplementationDataTypeElement` that fulfills all of the following conditions:

- The attribute `ImplementationDataTypeElement.arraySizeSemantics` shall not be defined.

- The attribute `ImplementationDataTypeElement.category` shall be set to `ARRAY`.
- The attribute `ImplementationDataTypeElement.arraySize` shall not be defined.
- The attribute `ImplementationDataTypeElement.arraySizeHandling` shall not be defined.

The `VSA Payload ImplementationDataTypeElement` shall immediately aggregate another `ImplementationDataTypeElement` that shall fulfill all of the following conditions:

- The attribute `ImplementationDataTypeElement.arraySizeSemantics` shall be set to the value `variableSize`.
- The attribute `ImplementationDataTypeElement.arraySize` shall be defined.
- The attribute `ImplementationDataTypeElement.arraySizeHandling` shall be set to the value `allIndicesSameArraySize`.

]

Please note that the `ImplementationDataTypeElement` aggregated by the `VSA Payload ImplementationDataTypeElement` can basically have any possible value of the attribute `category`.

[constr_1319] Profile `VSA_SQUARE` for `ImplementationDataType` [If the value of attribute `ImplementationDataType.dynamicArraySizeProfile` is set to `VSA_SQUARE`, the `ImplementationDataType` shall aggregate a `VSA Payload ImplementationDataTypeElement` that fulfills all of the the following conditions:

- The attribute `ImplementationDataTypeElement.arraySizeSemantics` shall not be defined.
- The attribute `ImplementationDataTypeElement.category` shall be set to the value `ARRAY`.
- The attribute `ImplementationDataTypeElement.arraySize` shall not be defined.
- The attribute `ImplementationDataTypeElement.arraySizeHandling` shall not be defined.

The `VSA Payload ImplementationDataTypeElement` shall immediately aggregate another `ImplementationDataTypeElement` (representing the first dimension) that shall fulfill all of the following conditions:

- The attribute `ImplementationDataTypeElement.arraySizeSemantics` shall be set to the value `variableSize`.

- The attribute `ImplementationDataTypeElement.category` shall be set to the value `ARRAY`.
- The attribute `ImplementationDataTypeElement.arraySize` shall not be defined.
- The attribute `ImplementationDataTypeElement.arraySizeHandling` shall be set to the value `inheritedFromArrayElementTypeSize`.

All **intermediate** `ImplementationDataTypeElements` in the aggregation chain that do not terminate the chain shall fulfill all of the following conditions:

- The attribute `ImplementationDataTypeElement.arraySizeSemantics` shall be set to the value `variableSize`.
- The attribute `ImplementationDataTypeElement.category` shall be set to the value `ARRAY`.
- The attribute `ImplementationDataTypeElement.arraySize` shall not be defined.
- The attribute `ImplementationDataTypeElement.arraySizeHandling` shall be set to the value `inheritedFromArrayElementTypeSize`.

The **terminating** `ImplementationDataTypeElement` in the aggregation chain shall fulfill all of the following conditions:

- The attribute `ImplementationDataTypeElement.arraySizeSemantics` shall be set to the value `variableSize`.
- The attribute `ImplementationDataTypeElement.arraySize` shall be defined.
- The attribute `ImplementationDataTypeElement.arraySizeHandling` shall be set to the value `allIndicesSameArraySize`.

⌋()

[constr_1320] Profile `VSA_RECTANGULAR` for `ImplementationDataType` ⌈ If the value of attribute `ImplementationDataType.dynamicArraySizeProfile` is set to `VSA_RECTANGULAR`, the `ImplementationDataType` shall aggregate a `VSA Payload ImplementationDataTypeElement` that fulfills all of the following conditions:

- The attribute `ImplementationDataTypeElement.arraySizeSemantics` shall not be defined.
- The attribute `ImplementationDataTypeElement.category` shall be set to the value `ARRAY`.
- The attribute `ImplementationDataTypeElement.arraySize` shall not be defined.

- The attribute `ImplementationDataTypeElement.arraySizeHandling` shall not be defined.

The `VSA Payload ImplementationDataTypeElement` shall immediately aggregate another `ImplementationDataTypeElement` (representing the first dimension) that shall fulfill all of the following conditions:

- The attribute `ImplementationDataTypeElement.category` shall be set to the value `ARRAY`.
- The attribute `ImplementationDataTypeElement.arraySizeSemantics` shall be set to the value `variableSize`.
- The attribute `ImplementationDataTypeElement.arraySize` shall be defined.
- The attribute `ImplementationDataTypeElement.arraySizeHandling` shall be set to the value `allIndicesSameArraySize`.

All **intermediate** `ImplementationDataTypeElements` in the aggregation chain that do not terminate the chain shall fulfill all of the following conditions:

- The attribute `ImplementationDataTypeElement.category` shall be set to the value `ARRAY`.
- The attribute `ImplementationDataTypeElement.arraySizeSemantics` shall be set to the value `variableSize`.
- The attribute `ImplementationDataTypeElement.arraySize` shall be defined.
- The attribute `ImplementationDataTypeElement.arraySizeHandling` shall be set to the value `allIndicesSameArraySize`.

The **terminating** `ImplementationDataTypeElement` in the aggregation chain shall fulfill all of the following conditions:

- The attribute `ImplementationDataTypeElement.arraySizeSemantics` shall be set to the value `variableSize`.
- The attribute `ImplementationDataTypeElement.arraySize` shall be defined.
- The attribute `ImplementationDataTypeElement.arraySizeHandling` shall be set to the value `allIndicesSameArraySize`.

]()

[constr_1321] Profile `VSA_FULLY_FLEXIBLE` for `ImplementationDataType` [If the value of attribute `ImplementationDataType.dynamicArraySizeProfile` is set to the value `VSA_FULLY_FLEXIBLE`, the `ImplementationDataType` shall aggregate a `VSA Payload ImplementationDataTypeElement` that fulfills all of the following conditions:

- The attribute `ImplementationDataTypeElement.arraySizeSemantics` shall not be defined.
- The attribute `ImplementationDataTypeElement.category` shall be set to the value `ARRAY`.
- The attribute `ImplementationDataTypeElement.arraySize` shall not be defined
- The attribute `ImplementationDataTypeElement.arraySizeHandling` shall not be defined.

The `VSA Payload ImplementationDataTypeElement` shall immediately aggregate another `ImplementationDataTypeElement` (representing the first dimension) that shall fulfill all of the following conditions:

- The attribute `ImplementationDataTypeElement.category` shall be set to `STRUCTURE`
- The attribute `ImplementationDataTypeElement.arraySizeSemantics` shall be set to the value `variableSize`.
- The attribute `ImplementationDataTypeElement.arraySize` shall be defined.
- The attribute `ImplementationDataTypeElement.arraySizeHandling` shall be set to the value `allIndicesDifferentArraySize`.

The `ImplementationDataTypeElement` shall aggregate another `ImplementationDataTypeElement` that fulfills the following conditions:

- The attribute `ImplementationDataTypeElement.arraySizeSemantics` shall not be defined.
- The attribute `ImplementationDataTypeElement.category` shall be set to the value `ARRAY`.
- The attribute `ImplementationDataTypeElement.arraySize` shall not be defined.
- The attribute `ImplementationDataTypeElement.arraySizeHandling` shall not be defined.

The **aggregation chain is continued** by a (possible empty) sequence of a pair of `ImplementationDataTypeElements` with the following characteristics:

- The first `ImplementationDataTypeElement` in the pair shall fulfill all of the following conditions:
 - The attribute `ImplementationDataTypeElement.category` shall be set to `STRUCTURE`.
 - The attribute `ImplementationDataTypeElement.arraySizeSemantics` shall be set to the value `variableSize`.

- The attribute `ImplementationDataTypeElement.arraySize` shall be defined.
- The attribute `ImplementationDataTypeElement.arraySizeHandling` shall be set to the value `allIndicesDifferentArraySize`.
- The second `ImplementationDataTypeElement` in the pair shall fulfill all of the following conditions:
 - The attribute `ImplementationDataTypeElement.arraySizeSemantics` shall not be defined.
 - The attribute `ImplementationDataTypeElement.category` shall be set to the value `ARRAY`.
 - The attribute `ImplementationDataTypeElement.arraySize` shall not be defined.
 - The attribute `ImplementationDataTypeElement.arraySizeHandling` shall not be defined.

The **terminating** `ImplementationDataTypeElement` in the aggregation chain shall fulfill all of the following conditions:

- The attribute `ImplementationDataTypeElement.arraySizeSemantics` shall be set to the value `variableSize`.
- The attribute `ImplementationDataTypeElement.arraySize` shall be defined.
- The attribute `ImplementationDataTypeElement.arraySizeHandling` shall be set to the value `allIndicesSameArraySize`.

]()

[constr_1396] Restriction for the value of attribute `category` for non-terminating `ImplementationDataTypeElements` taken to model a `Variable-Size Array Data Type` [The value of attribute `category` for non-terminating `ImplementationDataTypeElements` taken to model a `Variable-Size Array Data Type` shall **not** be set to `TYPE_REFERENCE`.]()

[constr_1322] `Size Indicator` for undefined `dynamicArraySizeProfile` [If the `ImplementationDataType.dynamicArraySizeProfile` does not exist but the `ImplementationDataType` is mapped to an `ApplicationArrayDataType` where the attribute `ApplicationArrayDataType.dynamicArraySizeProfile` exists, then the `ImplementationDataType` shall have the `category` `STRUCTURE`, representing a `Variable-Size Array Data Type` with `Size Indicator` enabled.]()

[TPS_SWCT_01617] Structure of an `ImplementationDataType` that represents a variable-sized array data type [The `ImplementationDataType` that represents

a `Variable-Size Array Data Type` shall have the `category` `STRUCTURE` that has two `subElements`.

The role of the `subElements` with the definition of a `Variable-Size Array Data Type` is defined by [TPS_SWCT_01618], [TPS_SWCT_01619], [TPS_SWCT_01620], and [TPS_SWCT_01621]. *|(RS_SWCT_03181)*

[TPS_SWCT_01618] Size Indicator for `dynamicArraySizeProfile` set to `VSA_LINEAR`, `VSA_SQUARE`, or `VSA_FULLY_FLEXIBLE` *|* If an `ImplementationDataType` is mapped to an `ApplicationArrayDataType` which has attribute `dynamicArraySizeProfile` set to the value `VSA_LINEAR`, `VSA_SQUARE` or `VSA_FULLY_FLEXIBLE`, the **first** `ImplementationDataType.subElement` shall be an integer large enough to hold the maximum number of valid elements of the variable size array (according to `maxArraySize`).

This is the `Size Indicator` which holds the current number of valid elements of the variable size array. *|(RS_SWCT_03181)*

[TPS_SWCT_01647] Size Indicator for `dynamicArraySizeProfile` set to `VSA_LINEAR`, `VSA_SQUARE`, or `VSA_FULLY_FLEXIBLE` if only `ImplementationDataType` is present *|* For each `ImplementationDataType` which has attribute `dynamicArraySizeProfile` set to the value `VSA_LINEAR`, `VSA_SQUARE`, or `VSA_FULLY_FLEXIBLE`, the first `ImplementationDataType.subElement` shall be an integer large enough to hold the maximum number of valid elements of the variable size array (according to `maxArraySize`).

This is the `Size Indicator` which holds the current number of valid elements of the `Variable-Size Array Data Type`. *|(RS_SWCT_03181)*

[TPS_SWCT_01619] Size Indicator for `dynamicArraySizeProfile` set to `VSA_RECTANGULAR` *|* If an `ImplementationDataType` is mapped to an `ApplicationArrayDataType` where the attribute `ApplicationArrayDataType.dynamicArraySizeProfile` exists and is set to the value `VSA_RECTANGULAR`, the **first** `ImplementationDataType.subElement` shall be a `ImplementationDataTypeElement` with the `category` set to `ARRAY` and the attribute `arraySize` set to a value equal to the number of the according dimension of the corresponding `ApplicationDataType`. *|(RS_SWCT_03181)*

[TPS_SWCT_01648] Size Indicator for `dynamicArraySizeProfile` set to `VSA_RECTANGULAR` if only `ImplementationDataType` is present *|* For each `ImplementationDataType` where the attribute `ImplementationDataType.dynamicArraySizeProfile` exists and is set to the value `VSA_RECTANGULAR`, the **first** `ImplementationDataType.subElement` shall be a `ImplementationDataTypeElement` with the `category` set to `ARRAY` and the attribute `arraySize` set to a value equal to the size of the according dimension of the rectangular array. *|(RS_SWCT_03181)*

[TPS_SWCT_01620] Size Indicator for `dynamicArraySizeProfile` set to `VSA_RECTANGULAR` *|* The elements of this `Size Indicator` array shall consist of

integers large enough to hold the maximum number of valid elements (according to `maxArraySize`). [\]\(RS_SWCT_03181\)](#)

This array holds the `Size Indicators` of all dimensions.

[TPS_SWCT_01621] Payload for `dynamicArraySizeProfile` [\[](#) If an `ImplementationDataType` is mapped to an `ApplicationArrayDataType` where the attribute `dynamicArraySizeProfile` exists, the **second** `ImplementationDataType.subElement` shall be an array which can hold the data of the variable size array with all dimensions defined for the `ApplicationDataType`.

The `category` shall be set to `ARRAY` and `arraySize` shall be set to `maxArraySize` of the corresponding `ApplicationArrayDataType`. [\]\(RS_SWCT_03181\)](#)

[TPS_SWCT_01649] Payload for `dynamicArraySizeProfile` if only `ImplementationDataType` is present [\[](#) Each `ImplementationDataType` where the attribute `dynamicArraySizeProfile` exists shall aggregate a second `ImplementationDataType.subElement` with the `category` set to `ARRAY`. [\]\(RS_SWCT_03181\)](#)

For examples, see Appendix [E.1](#).

An `ImplementationDataType` is also allowed to have `SwDataDefProps` (this feature is inherited from `AutosarDataType`), i.e. it can define various specific structural and semantical attributes. Table [5.39](#) shows which `SwDataDefProps` will be typically used here.

[TPS_SWCT_01257] `ImplementationDataType` or the aggregated `ImplementationDataTypeElements` do not form closed sets [\[](#) As figures [5.11](#) shows, an `ImplementationDataType` or the aggregated `ImplementationDataTypeElements` do not form closed sets but refer to further type definitions in one of four distinctive ways, depending on whether the type is implemented via a base type, a data or function pointer, or a reference to another implementation data type:

1. Reference to an underlying `SwBaseType` corresponds to `category VALUE`.
2. Reference to `BswModuleEntry` in `SwPointerTargetProps` corresponds to `category FUNCTION_REFERENCE`.
3. `SwDataDefProps` in `SwPointerTargetProps` corresponds to `category DATA_REFERENCE`.
4. Reference to another `ImplementationDataType` corresponds to `category TYPE_REFERENCE`.

[\]\(RS_SWCT_03217\)](#)

At the end, all the “leafs” of the complete tree formed by these references shall end up in `SwBaseTypes`. Figures [5.12](#), [5.13](#), and Figure [5.14](#) illustrate more examples about Typedefs and references.

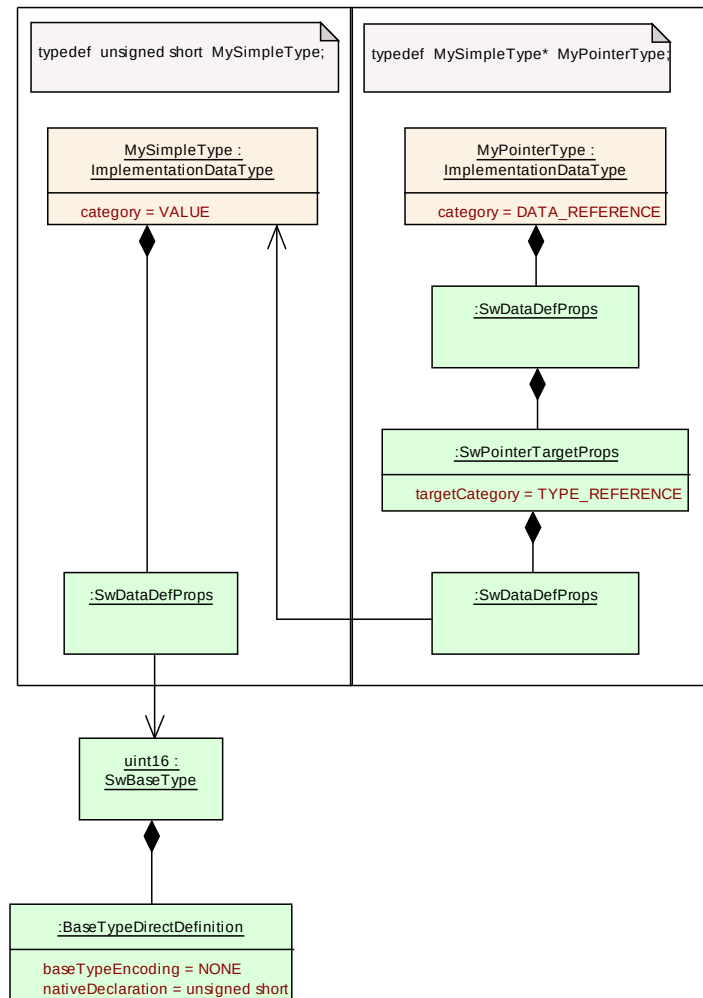


Figure 5.12: Example (1) for TypeDefs

[TPS_SWCT_01258] Definition of a pointer to data [The definition of a data pointer requires a special meta-class `SwPointerTargetProps` which aggregates another `SwDataDefProps`. This mechanism allows to describe the `category` and properties of the pointer object itself as well as the `category` and properties of its target data type.]([RS_SWCT_03217](#))

[constr_1177] Allowed `targetCategory` for `SwPointerTargetProps` [The value of `targetCategory` for `SwPointerTargetProps` can only be one of `TYPE_REFERENCE` or `FUNCTION_REFERENCE`. The only exception from this rule applies if the `swDataDefProps` owned by the `SwPointerTargetProps` refers to a `SwBaseType` with native type declaration `void`, in this case the value `VALUE` is also permitted.]()

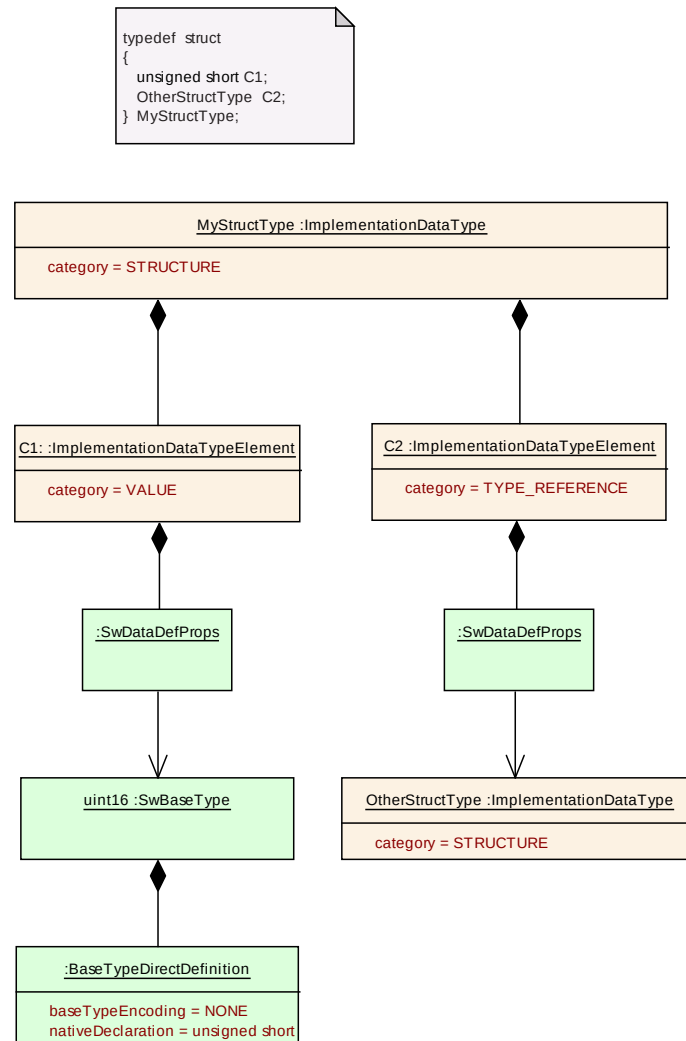


Figure 5.13: Example (2) for TypeDefs

As far as the AUTOSAR meta-model is concerned, a pointer to a pointer **could** in principle be implemented in two ways:

1. by defining an `ImplementationDataType` of category `DATA_REFERENCE` that aggregates `SwDataDefProps` in the role `swDataDefProps` that in turn aggregate `SwPointerTargetProps` in the role `swPointerTargetProps` with attribute `targetCategory` set to `TYPE_REFERENCE` that aggregates `SwDataDefProps` in the role `swDataDefProps` that references an `ImplementationDataType` of category `DATA_REFERENCE`.
2. by defining an `ImplementationDataType` of category `DATA_REFERENCE` that aggregates `SwDataDefProps` in the role `swDataDefProps` that in turn aggregate `SwPointerTargetProps` in the role `swPointerTargetProps` with attribute `targetCategory` set to `DATA_REFERENCE` (which is not allowed according to [constr_1177]) that in turn aggregates `SwDataDefProps` in the role `swDataDefProps` that aggregates `SwPointerTargetProps` in the role `swPointerTargetProps` that references an `ImplementationDataType` of category e.g. `VALUE`.

[constr_1254] Definition of a pointer to a pointer [AUTOSAR does **not** support the definition of a pointer to a pointer by defining an [ImplementationDataType](#) of category [DATA_REFERENCE](#) that aggregates [SwDataDefProps](#) in the role [swDataDefProps](#) that in turn aggregate [SwPointerTargetProps](#) in the role [swPointerTargetProps](#) with attribute [targetCategory](#) set to [DATA_REFERENCE](#) that in turn aggregates [SwDataDefProps](#) in the role [swDataDefProps](#) that aggregates [SwPointerTargetProps](#) in the role [swPointerTargetProps](#) that references an [ImplementationDataType](#) of category e.g. [VALUE](#).]()

For clarification, The AUTOSAR RTE does not support a definition of a pointer to a pointer by way of option 2 anyway. For all intents and purposes, [constr_1254] merely reflects this restriction on the level of AUTOSAR models. Option 1 (which is also featured in Figure 5.14) is the only viable way that is positively supported by the AUTOSAR RTE [2].

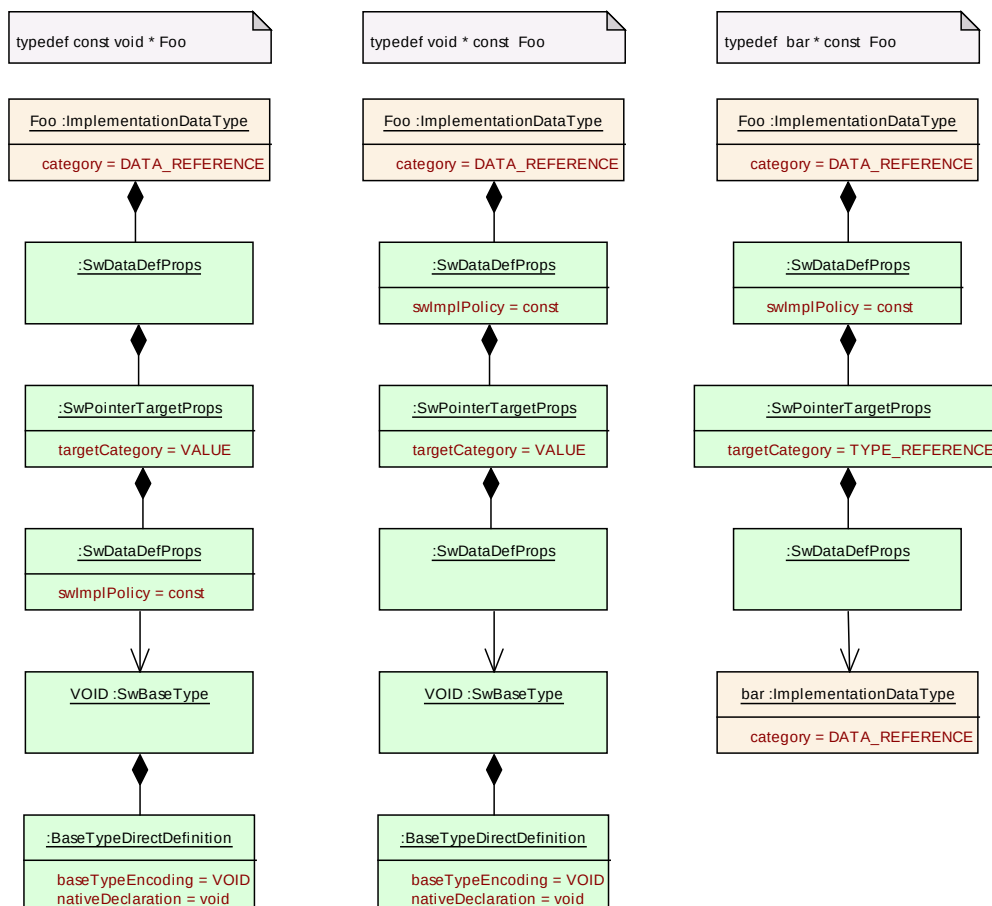


Figure 5.14: Example (3) for TypeDefs

[TPS_SWCT_01259] Definition of a pointer to a function [An [ImplementationDataType](#) or one of its sub-elements can also describe a function pointer. This completes its ability to declare all kinds of local data and of possible arguments used in library calls.

A function pointer is defined by the category [FUNCTION_REFERENCE](#) and the association [SwPointerTargetProps.functionPointerSignature](#) that refers to a

[BswModuleEntry](#). The latter essentially describes the signature of a function as explained in [7].]([RS_SWCT_03217](#))

Class	SwPointerTargetProps			
Package	M2::AUTOSARTemplates::CommonStructure::DataDefProperties			
Note	This element defines, that the data object (which is specified by the aggregating element) contains a reference to another data object or to a function in the CPU code. This corresponds to a pointer in the C-language. The attributes of this element describe the category and the detailed properties of the target which is either a data description or a function signature.			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
functionPointerSignature	BswModuleEntry	0..1	ref	The referenced BswModuleEntry serves as the signature of a function pointer definition. Primary use case: function pointer passed as argument to other function. Tags: xml.sequenceOffset=40
swDataDefProps	SwDataDefProps	0..1	aggr	The properties of the target data type. Tags: xml.sequenceOffset=30
targetCategory	Identifier	0..1	ref	This specifies the category of the target: • In case of a data pointer, it shall specify the category of the referenced data. • In case of a function pointer, it could be used to denote the category of the referenced BswModuleEntry. Since currently no categories for BswModuleEntry are defined it will be empty. Tags: xml.sequenceOffset=5

Table 5.21: SwPointerTargetProps

The allowed existence and multiplicity of all the attributes of [SwDataDefProps](#) and other properties depend on the [category](#) of the [ImplementationDataType](#).

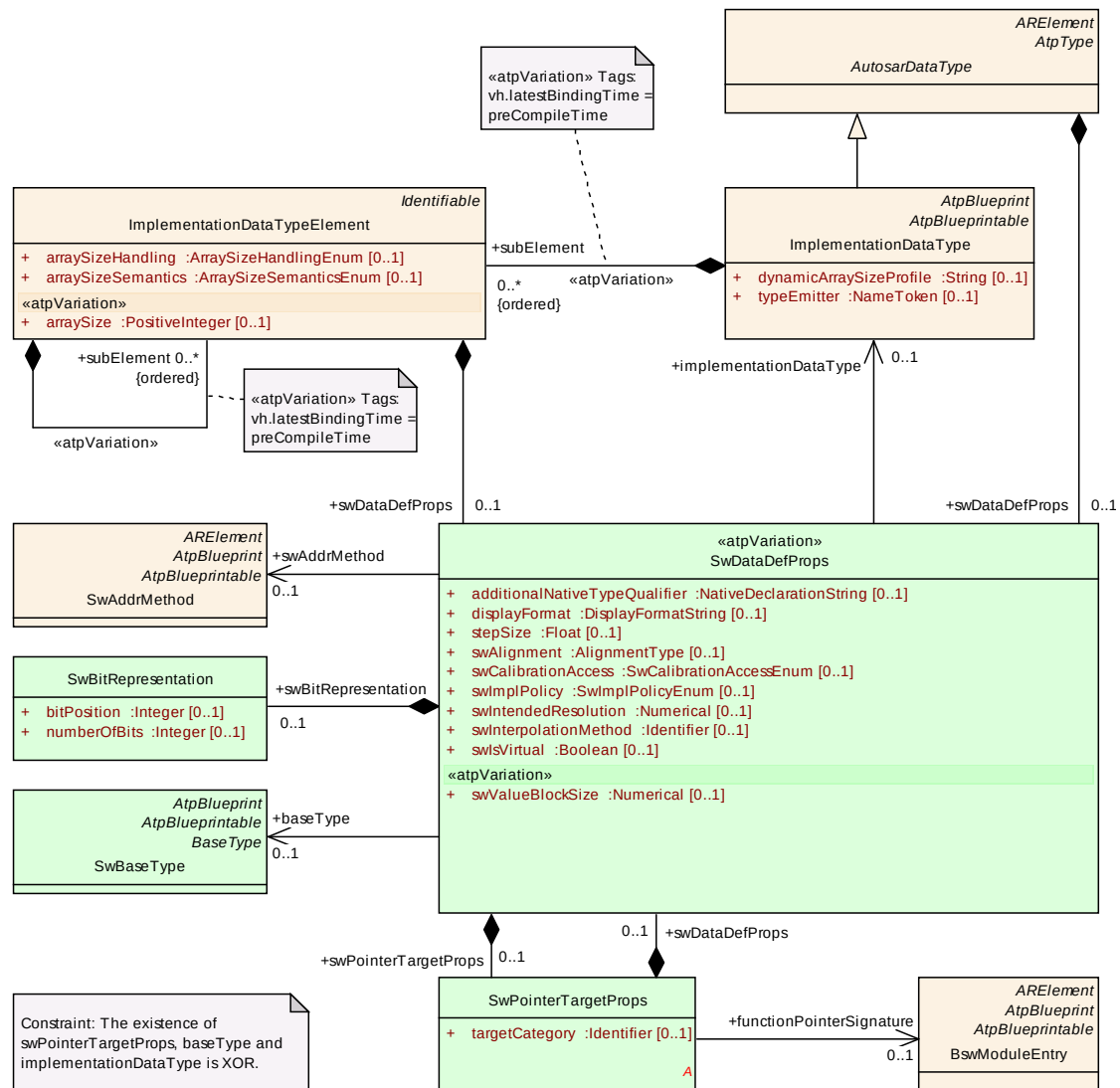


Figure 5.15: `SwDataDefProps` used in the context of `ImplementationDataType`

[constr_1178] Existence of attributes of `SwDataDefProps` in the context of `ImplementationDataType` [For the sake of removing possible sources of ambiguity, `SwDataDefProps` used in the context of `ImplementationDataType` can **only** have one of

- `baseType`
- `swPointerTargetProps`
- `implementationDataType`

]()

Please note that an `ImplementationDataType` manifests itself in the source code of an RTE into which a `DataPrototype` typed by the `ImplementationDataType` is deployed. This implies potential naming conflicts if `ImplementationDataTypes` that have identical `shortNames` are deployed into a specific RTE.

[TPS_SWCT_01194] **Symbolic name of an `ImplementationDataType`** [To mitigate this potential hazard it is possible to provide the `ImplementationDataType` along with an accompanying symbolic name that can be used for resolving the name clash. The symbolic name is provided by means of the attribute `symbol` of the meta-class `SymbolProps` owned by `ImplementationDataType` in the role `symbolProps` (for more information, please refer to Figure 5.11).]()

[TPS_SWCT_01441] **Nature of a `TYPE_REFERENCE`** [A type reference (formally represented by an `ImplementationDataType` of category `TYPE_REFERENCE`) implements a redirection to common `ImplementationDataTypes`.]()

[TPS_SWCT_01442] **`ImplementationDataType` of category `TYPE_REFERENCE` does not define own properties** [As long as an `ImplementationDataType` of category `TYPE_REFERENCE` does not define own properties the properties of the refined `ImplementationDataType` apply.]()

[TPS_SWCT_01443] **`ImplementationDataType` of category `TYPE_REFERENCE` overwrites properties of refined `ImplementationDataType`** [If an implementation data types of category `TYPE_REFERENCE` defines own properties (e.g. `CompuMethod`) this properties overwrite the properties of the refined `ImplementationDataType`.]()

As explained by [constr_1050], Compatibility checks of `ImplementationDataType` require a prior resolution of possible type references, i.e. the compatibility shall be checked on the resolved `ImplementationDataType`.

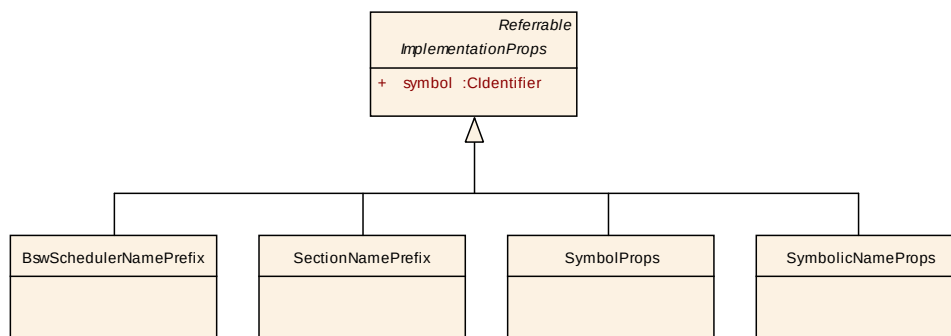


Figure 5.16: `ImplementationProps` and its subclasses

Class	ImplementationProps (abstract)			
Package	M2::AUTOSARTemplates::CommonStructure::Implementation			
Note	Defines a symbol to be used as (depending on the concrete case) either a complete replacement or a prefix when generating code artifacts.			
Base	ARObject, Referrable			
Attribute	Datatype	Mul.	Kind	Note
symbol	CIdentifier	1	ref	The symbol to be used as (depending on the concrete case) either a complete replacement or a prefix.

Table 5.22: ImplementationProps

Class	SymbolProps			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	This meta-class represents the ability to attach with the symbol attribute a symbolic name that is conform to C language requirements to another meta-class, e.g. AtomicSwComponentType, that is a potential subject to a name clash on the level of RTE source code.			
Base	ARObject, ImplementationProps , Referrable			
Attribute	Datatype	Mul.	Kind	Note
–	–	–	–	–

Table 5.23: SymbolProps

5.2.6 Base Type

[TPS_SWCT_01260] [SwBaseType](#) [[BaseType](#) is used to specify the basic level mentioned in chapter 5.1. In AUTOSAR, we use the meta-class [SwBaseType](#) which is derived from the abstract class [BaseType](#) due to other use cases for [BaseType](#) in ASAM HDO.]()

[TPS_SWCT_01261] **Use case for [SwBaseType](#)** [One use case for [SwBaseType](#) is to serve as input for the RTE generator. It will always appear at the “leaves” of data the types definitions which are relevant for RTE generation. It is used to generate the corresponding C-code typedef’s in case the attribute [BaseTypeDirectDefinition.nativeDeclaration](#) exists.]()

[constr_1010] **If [nativeDeclaration](#) does not exist** [If [nativeDeclaration](#) does not exist in the [SwBaseType](#) it is required that the [shortName](#) (e.g. “uint8”) of the corresponding [ImplementationDataType](#) is equal to a name of one of the Platform or Standard Types predefined in AUTOSAR code.]()

The consequence of [constr_1010] is that if the [nativeDeclaration](#) does not exist the RTE generator will **not** consider the [ImplementationDataType](#) for the generation of data type definitions.

Still, the compiler will positively be able to resolve the data type because it can fall back to the data type definitions contained in the header file for platform and standard data types that has to be included by regulation of the AUTOSAR standard.

Please note that [nativeDeclaration](#) shall yield a valid C data type symbol, whether this is done by a `typedef` or a by using the symbol⁸ of an integral data type is principally all the same.

Of course, using the symbol of an integral data type as the value of [nativeDeclaration](#) increases the odds that the enclosing [SwBaseType](#) can be used independently of the availability of the definition of a `typedef` that may or may not be available in a given context.

⁸The symbol does not necessarily have to consist of a single token, i.e. for all intents and purposes (for example) `unsigned char` is also considered the symbol of an integral C data type.

[TPS_SWCT_01563] Applicable values for [nativeDeclaration](#) [For the purpose of avoiding portability issues the value [nativeDeclaration](#) should only consist of the symbol of an integral C data type.]()

For more information on this refer to [22].

[TPS_SWCT_01263] Further use cases for [SwBaseType](#) [Within the basic software description, [SwBaseType](#) can be used (together with [ImplementationDataTypes](#)) for documentation or to specify variables for debugging. Furthermore, [SwBaseTypes](#) are required in the generation of support data for measurement and calibration tools. Please refer to [7] for details on these use cases.]()

A more detailed description of [BaseTypes](#) can also be found in *ASAM MCD 2 Harmonized Data Objects*.⁹

Class	BaseType (abstract)			
Package	M2::AUTOSARTemplates::CommonStructure::BaseTypes			
Note	This abstract meta-class represents the ability to specify a platform dependant base type.			
Base	ARElement , ARObject , CollectableElement , Identifiable , MultilanguageReferrable , PackageableElement , Referrable			
Attribute	Datatype	Mul.	Kind	Note
baseTypeDefinition	BaseTypeDefinition	1	aggr	This is the actual definition of the base type. Tags: xml.roleElement=false; xml.roleWrapperElement=false; xml.sequenceOffset=20; xml.typeElement=false; xml.typeWrapperElement=false

Table 5.24: BaseType

Class	SwBaseType			
Package	M2::AUTOSARTemplates::CommonStructure::BaseTypes			
Note	This meta-class represents a base type used within ECU software. Tags: atp.recommendedPackage=BaseTypes			
Base	ARElement , ARObject , AtpBlueprint , AtpBlueprintable , BaseType , CollectableElement , Identifiable , MultilanguageReferrable , PackageableElement , Referrable			
Attribute	Datatype	Mul.	Kind	Note
—	—	—	—	—

Table 5.25: SwBaseType

⁹The definition of *Harmonized Data Objects* can be retrieved from ASAM at www.asam.net. Access is limited to ASAM members.

Class	BaseTypeDefinition (abstract)			
Package	M2::AUTOSARTemplates::CommonStructure::BaseTypes			
Note	This meta-class represents the ability to define a basetype.			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
–	–	–	–	–

Table 5.26: BaseTypeDefinition

Class	BaseTypeDirectDefinition			
Package	M2::AUTOSARTemplates::CommonStructure::BaseTypes			
Note	This BaseType is defined directly (as opposite to a derived BaseType)			
Base	ARObject, BaseTypeDefinition			
Attribute	Datatype	Mul.	Kind	Note
baseType Encoding	BaseTypeEncodingString	1	attr	This specifies, how an object of the current BaseType is encoded, e.g. in an ECU within a message sequence. Tags: xml.sequenceOffset=90
baseType Size	PositiveInteger	0..1	attr	Describes the length of the data type specified in the container in bits. Tags: xml.sequenceOffset=70
byteOrder	ByteOrderEnum	0..1	attr	This attribute specifies the byte order of the base type. Tags: xml.sequenceOffset=110
maxBaseTypeSize	PositiveInteger	0..1	attr	Describes the maximum length of the BaseType in bits. Tags: xml.sequenceOffset=80
memAlignment	PositiveInteger	0..1	attr	This attribute describes the alignment of the memory object in bits. E.g. "8" specifies, that the object in question is aligned to a byte while "32" specifies that it is aligned four byte. If the value is set to "0" the meaning shall be interpreted as "unspecified". Tags: xml.sequenceOffset=100

Attribute	Datatype	Mul.	Kind	Note
nativeDeclaration	NativeDeclarationString	0..1	attr	This attribute describes the declaration of such a base type in the native programming language, primarily in the Programming language C. This can then be used by a code generator to include the necessary declarations into a header file. For example BaseType with <pre>shortName: "MyUnsignedInt" nativeDeclaration: "unsigned short"</pre> Results in <pre>typedef unsigned short MyUnsignedInt;</pre> If the attribute is not defined the referring ImplementationDataTypes will not be generated as a typedef by RTE. If a nativeDeclaration type is given it shall fulfill the characteristic given by baseTypeEncoding and baseTypeSize. This is required to ensure the consistent handling and interpretation by software components, RTE, COM and MCM systems. Tags: xml.sequenceOffset=120

Table 5.27: BaseTypeDirectDefinition

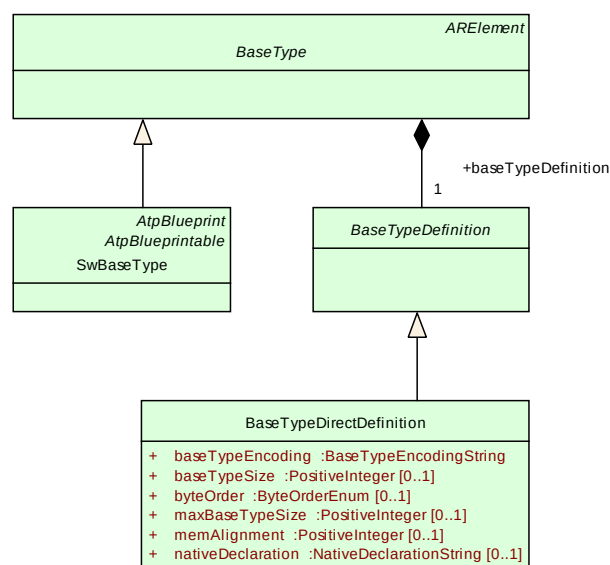


Figure 5.17: BaseType

Some additional hints to the properties of [SwBaseType](#):

- **[constr_1011] category of SwBaseType** [For the attribute `SwBaseType.category` only the values `FIXED_LENGTH` and `VARIABLE_LENGTH` are supported.]()
- **[constr_1012] Value of category is FIXED_LENGTH** [If the value of the attribute `SwBaseType.category` is set to `FIXED_LENGTH` then the attribute `baseTypeSize` shall be filled with content and attribute `maxBaseTypeSize` shall not exist.]()
- **[constr_1013] Value of category is VARIABLE_LENGTH** [If the value of the attribute `SwBaseType.category` is set to `VARIABLE_LENGTH` then the attribute `maxBaseTypeSize` shall be filled with content and attribute `baseTypeSize` shall not exist.]()
- **[TPS_SWCT_01444] Size of SwBaseType is specified in bits** [In both cases (mentioned in [constr_1012] and [constr_1013]) the size of `SwBaseType` is specified in bits.]()
- The attribute `baseTypeEncoding` specifies how the values of the base type are encoded.

[constr_1014] Supported value encodings for SwBaseType [The supported values for attribute `BaseTypeDirectDefinition.baseTypeEncoding` are:

- 1C: One's complement
- 2C: Two's complement
- BCD-P: Packed Binary Coded Decimals
- BCD-UP: Unpacked Binary Coded Decimals
- DSP-FRACTIONAL: Digital Signal Processor
- SM: Sign Magnitude
- IEEE754: floating point numbers
- ISO-8859-1: ASCII-Strings
- ISO-8859-2: ASCII-Strings
- WINDOWS-1252: ASCII-Strings
- UTF-8: UCS Transformation Format 8
- UTF-16: Character encoding for Unicode *code points* based on 16 bit *code units* [16]
- UCS-2: Universal Character Set 2
- NONE: Unsigned Integer
- VOID: corresponds to a void in C. The encoding is not formally specified here.

- **BOOLEAN:** This represents an unsigned integer to be interpreted as boolean. The value shall be interpreted as `true` if the value of the unsigned integer is 1 and it shall be interpreted as `false` if the value of the unsigned integer is 0.

A `CompuMethod` shall be referenced by the corresponding `Autosar-DataType` that implements the common sense behind the boolean concept, i.e. define a `TEXTTABLE` with two `CompuScales`: e.g. `true` → 1, `false` → 0.

⌋()

- **[TPS_SWCT_01262] `memAlignment` and `byteOrder` are platform-specific**
 [The value of attributes `BaseTypeDirectDefinition.memAlignment` and `BaseTypeDirectDefinition.byteOrder` is platform-specific and therefore should be set only in use cases where this is really needed.

These attributes shall be considered as optional.

If a `SwBaseType` is platform-specific then also the `ImplementationDataType` and software-component descriptions build on top of it become platform-specific.

⌋()

However, there are use cases for `SwBaseType` where this does not matter: especially the calibration support format which is generated in ECU-specific scope (and also contains `SwBaseType`, see [7]) could well be platform-specific.

Further regulations apply for the case that the value `UTF-16` is used for setting the attribute `BaseTypeDirectDefinition.baseTypeEncoding`:

[constr_1398] Existence of attributes of `BaseTypeDirectDefinition` [If the value of attribute `BaseTypeDirectDefinition.baseTypeEncoding` is set to `UTF-16` then the attribute `BaseTypeDirectDefinition.byteOrder` shall exist.

The only allowed values of `BaseTypeDirectDefinition.byteOrder` in this case are `mostSignificantByteFirst` and `mostSignificantByteLast` ⌋()

There is already predefined terminology (see [16]) existing that describes the two possible cases of byte orientation in a `UTF-16`-encoded string. The connection to this terminology is defined by [TPS_SWCT_01651] and [TPS_SWCT_01652].

[TPS_SWCT_01651] UTF-16BE [If the value of attribute `BaseTypeDirectDefinition.baseTypeEncoding` is set to `UTF-16` and the attribute `BaseTypeDirectDefinition.byteOrder` in this case are `mostSignificantByteFirst` then the `SwBaseType` corresponds to the definition of `UTF-16BE` according to the Unicode standard [16]. ⌋()

[TPS_SWCT_01652] UTF-16LE [If the value of attribute `BaseTypeDirectDefinition.baseTypeEncoding` is set to `UTF-16` and the attribute `BaseTypeDirectDefinition.byteOrder` in this case are `mostSignificantByteLast` then the `SwBaseType` corresponds to the definition of `UTF-16LE` according to the Unicode standard [16]. ⌋()

A further question that needs clarification is the usage of the so-called `Byte Order Mark` which allows (at run-time) for determining the actual byte order directly from the payload of a unicode string.

As AUTOSAR has means to formally and comprehensively define the byte order of any given `DataPrototype` that can hold a string at run time it is **not** necessary to support a further instrument that pretty much takes care of the same purpose.

[TPS_SWCT_01653] UTF-16-encoded strings are not allowed to start with a BOM
[If the value of attribute `BaseTypeDirectDefinition.baseTypeEncoding` is set to `UTF-16` then the value of a `DataPrototype` (which is effectively representing a string) is not allowed to start with a `Byte Order Mark` (BOM).]()

Please note that [TPS_SWCT_01653] removes a possible redundancy in the definition and execution of `UTF-16`-encoded strings.

The redundancy is not only regarded unnecessary but also **potentially dangerous** because it is not possible to check whether the definition is consistent with the execution at configuration time.

From the formal point of view, [TPS_SWCT_01653] does not represent an actual constraint although it is formulated as such.

However, an AUTOSAR tool would not be able to properly check the condition at configuration time and therefore this rule is published as a specification item.

5.2.7 Data Type Terminology

There are uses of data types that on the one hand need a handy term (because this kind of data type is used a lot) but on the other hand cannot easily be expressed in simple terms of meta-model elements (like `ApplicationDataType`).

Therefore, it is not an option to fully describe the characteristics of these kinds of data types precisely every time one of these is used. A definition of terminology is supposed to associate the mentioned kinds of data types with the term under which their use shall be paraphrased.

5.2.7.1 Primitive Type

In some cases it is necessary to constrain that applicability of data types to primitive C data types. It would be possible to describe the characteristics of eligible `Autosar-DataTypes` at every single place in an AUTOSAR specification where this specific limitation applies.

However, this may end up in lengthy and potentially inconsistent descriptions at different places within AUTOSAR specifications. Therefore, this chapter provides a canonical definition of a primitive data type that can be referred to from other places.

[TPS_SWCT_01564] Non-recursive definition of a primitive data type [An [AutosarDataType](#) is considered a primitive data type if the following conditions apply:

- it is an [ApplicationPrimitiveDataType](#) of category [VALUE](#) or [BOOLEAN](#)
- it is an [ImplementationDataType](#) of category [VALUE](#)

]()

[TPS_SWCT_01565] Recursive definition of a primitive data type [An [Autosar-DataType](#) is considered a primitive data type if the following conditions apply:

- it is an [AutosarDataType](#) according to [\[TPS_SWCT_01564\]](#)
- it is an [AutosarDataType](#) of category [TYPE_REFERENCE](#) that, after all type references have been resolved, boils down an [AutosarDataType](#) according to [\[TPS_SWCT_01564\]](#).

]()

5.2.7.2 Compound Primitive Data Type

[TPS_SWCT_01179] Compound Primitive Data Type [For clarification, a “compound primitive data type” is an [ApplicationPrimitiveDataType](#) of category [STRING](#), [CURVE](#), [MAP](#), [CUBOID](#), [CUBE_4](#), [CUBE_5](#), [COM_AXIS](#), [RES_AXIS](#), and [VAL_BLK](#).

This implies the existence of a [swRecordLayout](#) owned by the [swDataDefProps](#) of the [ApplicationPrimitiveDataType](#) that defines the mapping to a corresponding [ImplementationDataType](#).

The main characteristic of the “compound primitive data type” is that with respect to the application data type layer its data type is considered a primitive data type but when it comes to the implementation data type layer the type is implemented as a composite data type according to the applicable [SwRecordLayout](#).]([RS_SWCT_03216](#))

[TPS_SWCT_01486] [ApplicationPrimitiveDataType](#) of category [STRING](#) may have [invalidValue](#) [The only kind of [Compound Primitive Data Type](#) that is allowed to define an [invalidValue](#) is an [ApplicationPrimitive-DataType](#) of category [STRING](#).]([RS_SWCT_03216](#))

[constr_1241] [Compound Primitive Data Types](#) and [invalidValue](#) [[Compound Primitive Data Types](#) that have set the value of of category other than [STRING](#) shall **not** define [invalidValue](#).]()

5.2.7.3 Integral Primitive Type

The [SenderReceiverToSignalMapping](#) (see [11]) allows for the integral mapping of a piece of data to a single [SystemSignal](#). The specification of AUTOSAR COM

[21] imposes certain requirements on the characteristics of data that apply for the integral mapping.

[TPS_SWCT_01477] Integral Primitive Types [Data types that qualify for being used in the context of a The `SenderReceiverToSignalMapping` shall be called Integral Primitive Types.]([RS_SWCT_03218](#))

[constr_1229] category of ImplementationDataType boils down to VALUE [An `ImplementationDataType` qualifies as an `Integral Primitive Type` if and only if either

- its `category` is `VALUE` or `TYPE_REFERENCE` that eventually boils down to `VALUE` or
- its `category` is `ARRAY` and it has only one `subElement` and one of the following conditions applies:
 - `subElement.category` is set to `VALUE` or `TYPE_REFERENCE` that eventually boils down to `VALUE` and the `subElement` refers to a `SwBaseType` where `baseTypeSize` or `maxBaseTypeSize` is set to the value 8 and the `baseTypeEncoding` is set to `NONE`.
 - `subElement.category` is set to `TYPE_REFERENCE` and the `swDataDefProps.implementationDataType` literally represents the Platform Data Type named “uint8”.
 - `subElement.category` is set to `TYPE_REFERENCE` and the attribute `swDataDefProps.implementationDataType.shortName` is set to “uint8” and `swDataDefProps.baseType.baseTypeDefinition.nativeDeclaration` does not exist.

]()

[constr_1230] ApplicationDataType that qualifies for Integral Primitive Type [An `ApplicationDataType` qualifies as an `Integral Primitive Type` if and only if all of the following conditions apply:

- `ApplicationDataType.category` is set to `BOOLEAN`, `VALUE`, `STRING`, or `ARRAY`
- in the applicable scope a `DataTypeMap` is available that refers to the given `ApplicationDataType`
- the found `DataTypeMap` refers to an `ImplementationDataType` that fulfills the requirements of [\[constr_1229\]](#)

]()

5.2.7.4 Variable-Size Array Data Type

The definition of and further explanation regarding the term [Variable-Size Array Data Type](#) can be found in chapter [2.8](#).

5.3 Data Prototypes

5.3.1 Overview

[TPS_SWCT_01264] Data prototypes implement a role of a data type [Generally speaking, a data prototype represents the implementation of a role of a data type within the definition of another data type, e.g. a “typed” data object declared within a software component or a port interface. This means formally that it has an is-of-type relation to a data type and is usually aggregated by another element, e.g. the internal behavior or a port interface.]()

In the meta-model, various kinds of data prototypes are derived from the abstract [DataPrototype](#) as shown in figure [5.18](#). The reason for the introduction of this hierarchy was the distinction between [AutosarDataPrototype](#) (which can be used for the application and implementation types as well) and [ApplicationCompositeElementDataPrototype](#) (which is restricted to be used within the application types).

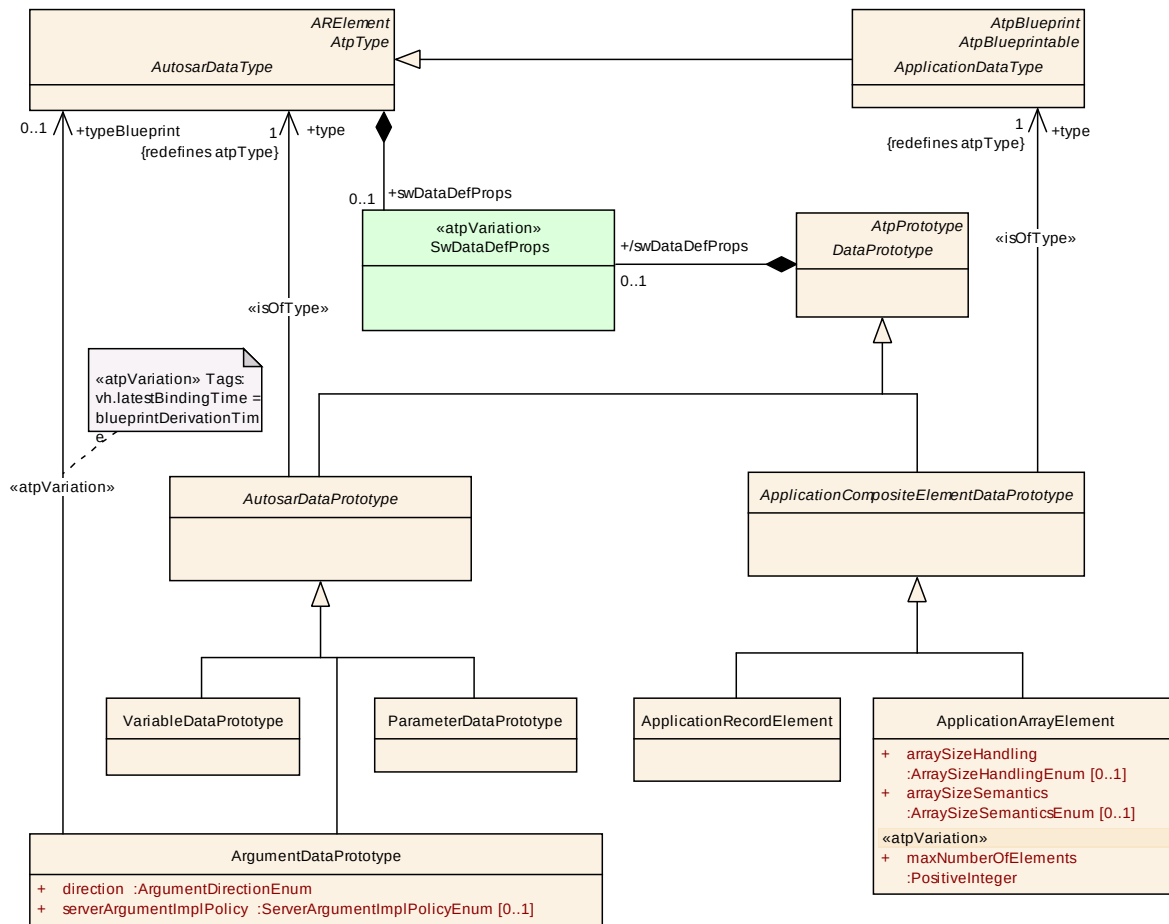


Figure 5.18: Data Prototypes Overview

Class	DataPrototype (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::DataPrototypes			
Note	Base class for prototypical roles of any data type.			
Base	ARObject, AtpFeature, AtpPrototype, Identifiable, MultilanguageReferrable, Referrable			
Attribute	Datatype	Mul.	Kind	Note
swDataDef Props	SwDataDefProps	0..1	aggr	This property allows to specify data definition properties which apply on data prototype level.

Table 5.28: DataPrototype

Class	AutosarDataPrototype (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::DataPrototypes			
Note	Base class for prototypical roles of an AutosarDataType.			
Base	ARObject, AtpFeature, AtpPrototype, DataPrototype, Identifiable, MultilanguageReferrable, Referrable			
Attribute	Datatype	Mul.	Kind	Note
type	AutosarDataType	1	tref	This represents the corresponding data type. Stereotypes: isOfType

Table 5.29: AutosarDataPrototype

Class	ApplicationCompositeElementDataPrototype (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::DataPrototypes			
Note	This class represents a data prototype which is aggregated within a composite application data type (record or array). It is introduced to provide a better distinction between target and context in instanceRefs.			
Base	ARObject, AtpFeature, AtpPrototype, DataPrototype , Identifiable , Multilanguage Referrable, Referrable			
Attribute	Datatype	Mul.	Kind	Note
type	ApplicationData Type	1	tref	This represents the corresponding data type. Stereotypes: isOfType

Table 5.30: ApplicationCompositeElementDataPrototype

Because these [DataPrototypes](#) are modeled as own meta-classes it is possible to define own attributes for them (on M2) which (in the M1 model) could extend or constrain the attribute values already set via the corresponding data type.

[TPS_SWCT_01265] [DataPrototype](#) aggregates an own set of [SwDataDefProps](#) [This mechanism is used here in the way that [DataPrototype](#) aggregates an own set of [SwDataDefProps](#). Thus each kind of [DataPrototype](#) has the ability to extend or even overwrite the [SwDataDefProps](#) already defined by its [ApplicationDataType](#) or [ImplementationDataType](#).

This mechanism, if carefully applied, allows for a better reuse of data types because they can be kept free of the properties which vary according to the context or are defined in later project phases. Chapter 5.4 describes more details on this.]()

[TPS_SWCT_01445] Applicability of [SwDataDefProps](#) for [DataPrototypes](#) [The applicability of [SwDataDefProps](#) for [DataPrototypes](#) shall follow the same rules as for the [categorys](#) of the corresponding [AutosarDataTypes](#) (see table 5.8).]()

Further information can be found in table 5.31 and table 5.32.

Please note that table 5.31 does not include the [ApplicationRecordElement](#) and [ApplicationArrayElement](#) because these specializations of [ApplicationCompositeElementDataPrototype](#) are already part of table 5.8. The same applies for table 5.32 which does not include the [ImplementationDataTypeElement](#).

	Root Elem.			Attribute Existence per Category										
	DataPrototype	InstantiationDataDefProps	ParameterAccess	VALUE	VAL_BLK	STRUCTURE	ARRAY	STRING	BOOLEAN	COM_AXIS	RES_AXIS	CURVE	MAP	CUBOID
Attributes of SwDataDefProps														
additionalNativeTypeQualifier														

	Root Elem.			Attribute Existence per Category												
	DataPrototype	InstantiationDataDefProps	ParameterAccess	VALUE	VAL_BLK	STRUCTURE	ARRAY	STRING	BOOLEAN	COM_AXIS	RES_AXIS	CURVE	MAP	CUBOID	CUBE_4	CUBE_5
Attributes of SwDataDefProps																
annotation	x	x	x	*	*	*	*	*	*	*	*	*	*	*	*	*
baseType																
compuMethod																
dataConstr	x	x		0..1	0..1				0..1			0..1	0..1	0..1	0..1	0..1
displayFormat	x	x		0..1	0..1			0..1	0..1			0..1	0..1	0..1	0..1	0..1
implementationDataType																
invalidValue																
stepSize	x	x	x	0..1	0..1					0..1	0..1	0..1	0..1	0..1	0..1	0..1
swAddrMethod	x	x		0..1	0..1	0..1	0..1	0..1	0..1	0..1	0..1	0..1	0..1	0..1	0..1	0..1
swAlignment	x	x		0..1	0..1	0..1	0..1	0..1	0..1	0..1	0..1	0..1	0..1	0..1	0..1	0..1
swBitRepresentation																
swCalibrationAccess	x	x		0..1	0..1	0..1	0..1	0..1	0..1	0..1	0..1	0..1	0..1	0..1	0..1	0..1
swCalprmAxisSet																
swCalprmAxisSet.swCalprmAxis/SwAxisGrouped.swCalprmRef		x	x									0..1	0..1	0..1	0..1	0..1
swCalprmAxisSet.swCalprmAxis/SwAxisIndividual.swVariableRef		x	x							0..1	0..1	0..1	0..1	0..1	0..1	0..1
swCalprmAxisSet.swCalprmAxis/SwAxisGrouped.sharedAxisType																
swCalprmAxisSet.swCalprmAxis/SwAxisIndividual.inputVariableType																
swCalprmAxisSet.swCalprmAxis/SwAxisIndividual.unit																
swCalprmAxisSet.swCalprmAxis.baseType																
swComparisonVariable			x									0..*	0..*	0..*	0..*	0..*
swDataDependency	x	x		0..1								0..1	0..1	0..1	0..1	0..1
swHostVariable																
swImplPolicy	x			0..1	0..1	0..1	0..1	0..1	0..1	0..1	0..1	0..1	0..1	0..1	0..1	0..1
swIntendedResolution																
swInterpolationMethod	x	x	x	0..1						0..1	0..1	0..1	0..1	0..1	0..1	0..1
swIsVirtual	x	x		0..1					0..1			0..1	0..1	0..1	0..1	0..1
swPointerTargetProps																
swRecordLayout																
swRefreshTiming	x	x		0..1	0..1			0..1	0..1							
swTextProps																
swValueBlockSize																
unit																
valueAxisDataType																

Table 5.31: Allowed Attributes vs. [category](#) for [DataPrototypes](#) typed by Application Data Types

[constr_1289] Allowed Attributes vs. **category** for **DataPrototypes** typed by **ApplicationDataTypes** [The allowed values of Attributes per **category** for **DataPrototypes** typed by **ApplicationDataTypes** are documented in table 5.31.]()

	Root Element			Attribute Existence per Category						
	DataPrototype	InstantiationDataDefProps	ParameterAccess	VALUE	DATA_REFERENCE	FUNCTION_REFERENCE	TYPE_REFERENCE	STRUCTURE	UNION	ARRAY
Attributes of SwDataDefProps										
additionalNativeTypeQualifier				*	*	*	*	*	*	*
annotation	x	x	x	*	*	*	*	*	*	*
baseType										
compuMethod										
dataConstr	x	x		0..1			0..1			
displayFormat	x	x		0..1			0..1	0..1	0..1	0..1
implementationDataType										
invalidValue				0..1						
stepSize	x	x		0..1						
swAddrMethod	x	x		0..1	0..1	0..1	0..1	0..1	0..1	0..1
swAlignment	x	x		0..1	0..1	0..1	0..1	0..1	0..1	0..1
swBitRepresentation										
swCalibrationAccess	x	x		0..1			0..1	0..1	0..1	0..1
swCalprmAxisSet										
swComparisonVariable										
swDataDependency										
swHostVariable										
swImplPolicy	x			0..1	0..1	0..1	0..1	0..1	0..1	0..1
swIntendedResolution										
swInterpolationMethod										
swIsVirtual										
swPointerTargetProps										
swPointerTargetProps.swDataDefProps										
swPointerTargetProps.functionPointerSignature										
swRecordLayout										
swRefreshTiming	x	x		0..1			0..1	0..1	0..1	0..1
swTextProps										
swValueBlockSize										
unit										
valueAxisDataType										

Table 5.32: Allowed Attributes vs. **category** for **DataPrototypes** typed by **ImplementationDataTypes**

[constr_1288] Allowed Attributes vs. **category** for **DataPrototypes** typed by **ImplementationDataTypes** [The allowed values per **category** for **DataPrototypes** typed by **ImplementationDataTypes** are documented in table 5.32.]()

[TPS_SWCT_01266] Three non-abstract classes derived from [AutosarDataPrototype](#) [There are three non-abstract classes derived from [AutosarDataPrototype](#) which reflect the main use cases in the SWC-Template:

- Operation arguments ([ArgumentDataPrototype](#)) in a client-server interface.
- Variables ([VariableDataPrototype](#)) which are changed by the application software at runtime.
- Parameters ([ParameterDataPrototype](#)) which are constant (except for calibration access) from the application point of view.

]()

Class	ArgumentDataPrototype			
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
Note	An argument of an operation, much like a data element, but also carries direction information and is owned by a particular ClientServerOperation.			
Base	ARObject,AtpFeature,AtpPrototype, AutosarDataPrototype , DataPrototype , Identifiable , MultilanguageReferrable , Referrable			
Attribute	Datatype	Mul.	Kind	Note
direction	ArgumentDirectionEnum	1	attr	This attribute specifies the direction of the argument prototype.
serverArgumentImplPolicy	ServerArgumentImplPolicyEnum	0..1	attr	This defines how the argument type of the servers RunnableEntity is implemented. If the attribute is not defined this has the same semantics as if the attribute is set to the value useArgumentType for primitive arguments and structures and to the value useArrayBaseType for arrays.
typeBlueprint	AutosarDataPrototype	0..1	ref	This allows to denote the intended type within blueprints. It shall be replaced by a proper type when deriving Interfaces from the Blueprint. Stereotypes: atpVariation Tags: vh.latestBindingTime=blueprintDerivationTime

Table 5.33: ArgumentDataPrototype

Class	VariableDataPrototype			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::DataPrototypes			
Note	A VariableDataPrototype is used to contain values in an ECU application. This means that most likely a VariableDataPrototype allocates "static" memory on the ECU. In some cases optimization strategies might lead to a situation where the memory allocation can be avoided. In particular, the value of a VariableDataPrototype is likely to change as the ECU on which it is used executes.			
Base	ARObject, AtpFeature, AtpPrototype, AutosarDataPrototype, DataPrototype, Identifiable, MultilanguageReferrable, Referrable			
Attribute	Datatype	Mul.	Kind	Note
initValue	ValueSpecification	0..1	aggr	Specifies initial value(s) of the VariableDataPrototype

Table 5.34: VariableDataPrototype

Class	ParameterDataPrototype			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::DataPrototypes			
Note	A parameter element used for parameter interface and internal behavior, supporting signal like parameter and characteristic value communication patterns and parameter and characteristic value definition.			
Base	ARObject, AtpFeature, AtpPrototype, AutosarDataPrototype, DataPrototype, Identifiable, MultilanguageReferrable, Referrable			
Attribute	Datatype	Mul.	Kind	Note
initValue	ValueSpecification	0..1	aggr	Specifies initial value(s) of the ParameterDataPrototype

Table 5.35: ParameterDataPrototype

[TPS_SWCT_01267] **DataPrototype can be aggregated in different roles** [Note that even though the meta-classes VariableDataPrototype and ParameterDataPrototype already express specific use cases of the underlying data type the same DataPrototype can still be aggregated in different roles, e.g. in the SwcInternalBehavior to express different methods how to access it.]()

An example is the aggregation of VariableDataPrototype by SwcInternalBehavior in the roles of either implicitInterRunnableVariable or explicitInterRunnableVariable. Find more information concerning these use cases in chapter 7.

[TPS_SWCT_01268] **Definition of initValue for a VariableDataPrototype or a ParameterDataPrototype** [It is possible to assign an initValue for both a VariableDataPrototype and a ParameterDataPrototype. This aspect is sketched in Figure 5.19.]()

[TPS_SWCT_01269] **In PortInterfaces, initial values defined for DataPrototypes are ignored** [These initValues have no meaning for DataPrototypes within PortInterfaces because in this case a more specific definition of initial values via the so-called ComSpec is required, see chapter 4.5.]()

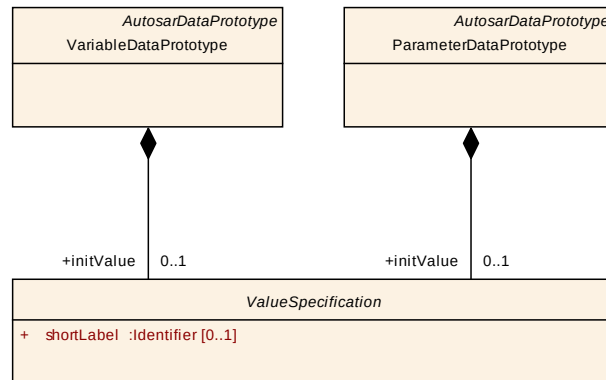


Figure 5.19: Initial value for **AutosarDataPrototypes**

Find more information about the interpretation of `initValue` in section 5.7.

5.3.2 Reference to Data Prototypes

This chapter explains the various patterns for referencing **DataPrototypes**.

[TPS_SWCT_01446] References to a **DataPrototype may or may not imply the necessity for using an `instanceRef`** [As references to a **DataPrototype** may or may not imply the necessity for using an `instanceRef` this would mean that in some places the meta-model would have to implement both variants depending on the use case. To avoid this, AUTOSAR defines a unified reference implementation for **VariableDataPrototypes** and **ParameterDataPrototypes**.]()

[TPS_SWCT_01270] **AutosarVariableRef** [With the advent of **AutosarVariableRef** it is possible to implement a uniform reference to a **VariableDataPrototype** that covers all foreseen use cases:

- Reference to a **localVariable**, no **AtpInstanceRef** required.
- Reference to an **autosarVariable** (which involves an **AtpInstanceRef**).
- Reference to the internal structure of a **VariableDataPrototype** implemented using a composite **ImplementationDataType**.

]()

Class	AutosarVariableRef			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::Data Elements			
Note	This class represents a reference to a variable within AUTOSAR which can be one of the following use cases: localVariable: - localVariable which is used as whole (e.g. InterRunnableVariable, inputValue for curve) autosarVariable: - a variable provided via Port which is used as whole (e.g. dataAccesspoints) - an element inside of a composite local variable typed by ApplicationDatatype (e.g. inputValue for a curve) - an element inside of a composite variable provided via Port and typed by ApplicationDatatype (e.g. inputValue for a curve) autosarVariableInImplDatatype: - an element inside of a composite local variable typed by ImplementationDatatype (e.g. nvramData mapping) - an element inside of a composite variable provided via Port and typed by ImplementationDatatype (e.g. inputValue for a curve)			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
autosarVariable	DataPrototype	0..1	iref	This references a variable which is provided by a port and/or which is part of a CompositeDataType.
autosarVariableInImplDatatype	ArVariableInImplementationDataInstanceRef	0..1	aggr	This is used if the target variable is inside of variableDataPrototype typed by an ImplementationDataType.
localVariable	VariableDataPrototype	0..1	ref	This reference is used if the variable is local to the current component. It would also be possible to use the instance reference here. Such an instance ref would not have a contextElement, since the current instance is the context. But the local instance is a special case which may provide further optimization. Therefore an explicit reference is provided for this case.

Table 5.36: AutosarVariableRef

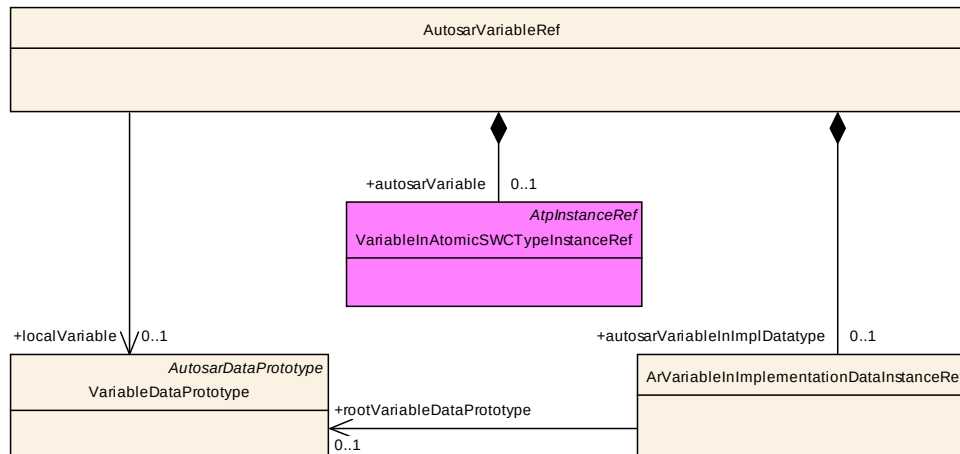


Figure 5.20: Implementation of **AutosarVariableRef**

Class	ArVariableInImplementationDataInstanceRef			
Package	M2::AUTOSAR Templates::SWComponentTemplate::SwcInternalBehavior::Data Elements			
Note	This class represents the ability to navigate into a data element inside of an VariableDataPrototype which is typed by an ImplementationDatatype. Note that it shall not be used if the target is the VariableDataPrototype itself (e.g. if its a primitive). Note that this class follows the pattern of an InstanceRef but is not implemented based on the abstract classes because the ImplementationDataType isn't either, especially because ImplementationDataTypeElement isn't derived from AtpPrototype.			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
contextDataPrototype (ordered)	ImplementationDataTypeElement	*	ref	This is a context in case there are subelements with explicit types. The reference has to be ordered to properly reflect the nested structure. Tags: xml.sequenceOffset=30
portPrototype	PortPrototype	0..1	ref	This is the port providing/receiving the root of the variable Tags: xml.sequenceOffset=10
rootVariableDataPrototype	VariableDataPrototype	0..1	ref	This refers to the variableDataPrototype which is typed by the implementationDatatype in which the target can be found. Tags: xml.sequenceOffset=20
targetDataPrototype	ImplementationDataTypeElement	1	ref	This is a context in case there are subelements with explicit types. Tags: xml.sequenceOffset=40

Table 5.37: ArVariableInImplementationDataInstanceRef

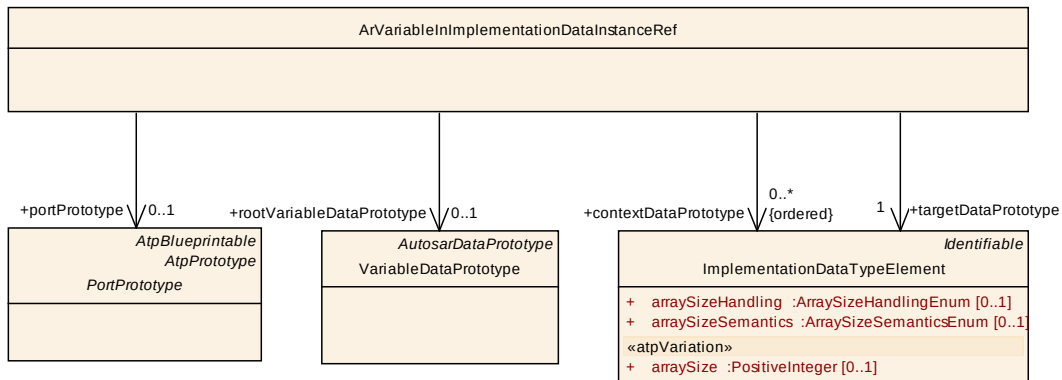


Figure 5.21: Implementation of **ArVariableInImplementationDataInstanceRef**

[constr_2536] Target of an **autosarVariable in **AutosarVariableRef** shall refer to a variable** [The target of **autosarVariable** (which in fact is an instance ref) in **AutosarVariableRef** shall either be or be nested in **VariableDataPrototype**. This means that the target shall either be a **VariableDataPrototype** or an **ApplicationCompositeElementDataPrototype** that in turn is owned by a **VariableDataPrototype**.]()

[TPS_SWCT_01271] **AutosarParameterRef** [With the advent of **AutosarParameterRef** it is possible to implement a uniform reference to a **ParameterDataPrototype** that covers all foreseen use cases:

- Reference to a **localParameter**, no **AtpInstanceRef** required.
- Reference to an **autosarParameter** (which involves an **AtpInstanceRef**).

]()

Please note that there is a very limited amount of use-cases available where the **AutosarParameterRef** can (with the active consent of the AUTOSAR standard) reference a **VariableDataPrototype**.

[constr_1173] Applicability of **AutosarParameterRef referencing a **VariableDataPrototype**** [A reference from **AutosarParameterRef** to **VariableDataPrototype** is **only** applicable if the **AutosarParameterRef** is used in the context of **SwAxisGrouped**.]()

For example, the use case referenced in **[constr_1173]** applies if it is required to store a grouped axis in a variable in order to adapt the axis during run-time of the ECU by a dedicated algorithm. Note that in all cases where **[constr_1173]** does not apply **[constr_2535]** shall be fulfilled.

Class	AutosarParameterRef			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::Data Elements			
Note	This class represents a reference to a parameter within AUTOSAR which can be one of the following use cases: localParameter: - localParameter which is used as whole (e.g. sharedAxis for curve) autosarVariable: - a parameter provided via PortPrototype which is used as whole (e.g. parameterAccess) - an element inside of a composite local parameter typed by ApplicationDatatype (e.g. sharedAxis for a curve) - an element inside of a composite parameter provided via Port and typed by ApplicationDatatype (e.g. sharedAxis for a curve) autosarParameterInImplDatatype: - an element inside of a composite local parameter typed by ImplementationDatatype - an element inside of a composite parameter provided via PortPrototype and typed by ImplementationDatatype			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
autosarParameter	DataPrototype	0..1	iref	This instance reference is used if the callibration parameter is either imported via a port or is part of a composite data structure.
localParameter	DataPrototype	0..1	ref	In the majority of cases this reference goes to ParameterDataPrototypes rather than VariableDataPrototypes. Pointing the reference to a VariableDataPrototype is limited to special use cases, e.g. if the AutosarParameterRef is used in the context of an SwAxisGrouped. This reference is used if the arParameter is local to the current component. Of course, it would technically also be feasible to use an InstanceRef for this case. However, the InstanceRef would not have a contextElement (because the cureent instance is the context). Hence, the local instance is a special case which may provide further optimization. Therefore an explicit reference is provided for this case.

Table 5.38: AutosarParameterRef

[constr_2535] Target of an `autosarParameter` in `AutosarParameterRef` shall refer to a parameter [Except for the specifically described cases where [\[constr_1173\]](#) applies the target of `autosarParameter` (which in fact is an instance ref) in `AutosarParameterRef` shall either be or be nested in `ParameterDataPrototype`. This means that the target shall either be a `ParameterDataPrototype` or an `ApplicationCompositeElementDataPrototype` that in turn is owned by a `ParameterDataPrototype`.]()

[constr_1161] Applicability of the `index` attribute of `Ref` [The `index` attribute of `Ref` is limited to a given set if use cases as there are:

- `McDataInstance.instanceInMemory`
- `AutosarVariableRef`
- `AutosarParameterRef`
- `FlatInstanceDescriptor / AnyInstanceRef`

]()

The implementation of the `AtpInstanceRefs` for `AutosarVariableRef` and `AutosarParameterRef` probably needs some clarification regarding the references to `DataPrototypes`.

[TPS_SWCT_01374] Implementation of `AutosarParameterRef` [The reference to `rootParameterDataPrototype` is **not** redundant. It is required for identifying the `autosarParameter` itself in a `ParameterInterface` **if and only if** the `AutosarDataType` of the `autosarParameter` is a composite data type. If the `AutosarDataType` was a primitive data type the `targetDataPrototype` reference is the **only** reference required.]()

As explained before, the implementation of `AutosarParameterRef` in a specific case is subject to [\[constr_1173\]](#).

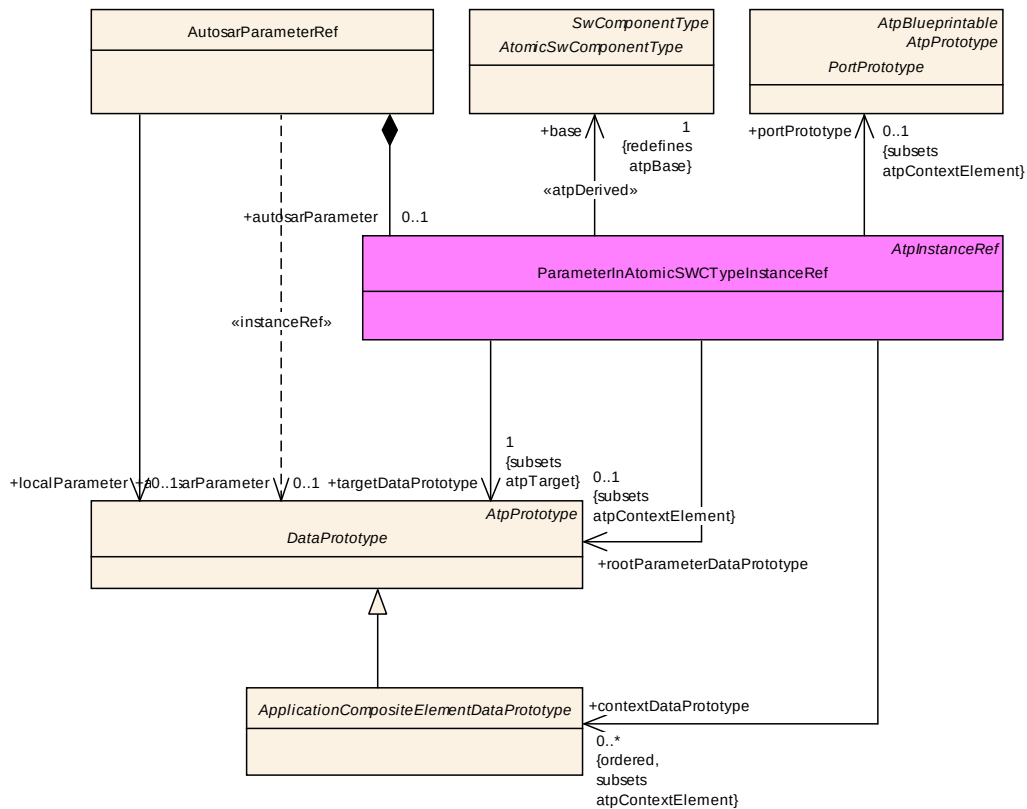


Figure 5.22: Implementation of the InstanceRef for **AutosarParameterRef**

[TPS_SWCT_01375] Implementation of **AutosarVariableRef** [The reference to **rootVariableDataPrototype** is **not** redundant. It is required for identifying the **autosarVariable** itself in a **SenderReceiverInterface** or **NvDataInterface** **if and only if** the **AutosarDataType** of the **autosarVariable** is a composite data type. If the **AutosarDataType** was a primitive data type the **targetDataPrototype** reference is the **only** reference required.]()

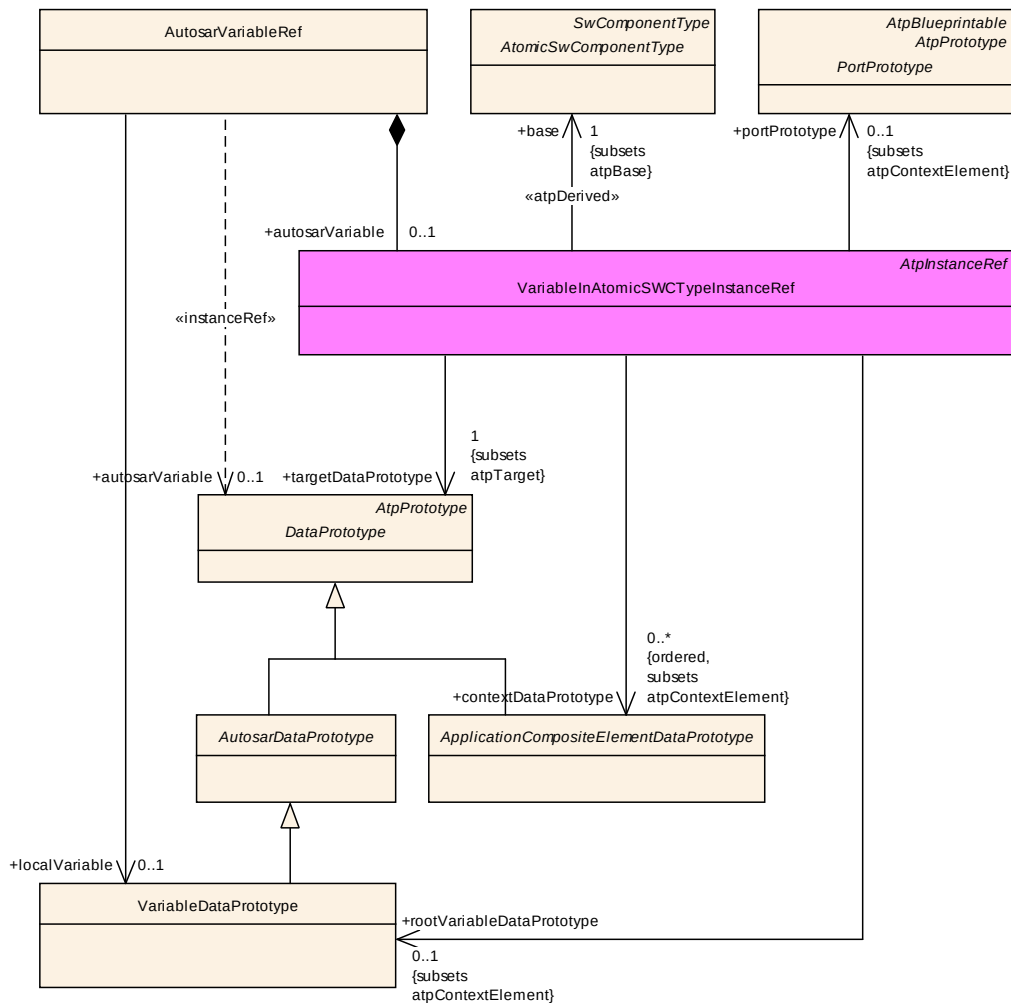


Figure 5.23: Implementation of the InstanceRef for **AutosarVariableRef**

5.4 Properties of Data Definitions

5.4.1 Overview

As it has already been shown in the previous chapters, various properties and associations can be attached to the definition of data types as well as prototypes. These are described by the meta-class **SwDataDefProps** which covers all properties of a particular data object under various aspects.

In general, the properties specified within **SwDataDefProps** may apply to all kind of data declared within the software-component template and within the basic software module description template as well, e.g. component local data, data used for communication, data used for measurement as well as for calibration.

However, there are constraints for the attributes depending on the role of the data:

[constr_1015] Prioritization of `SwDataDefProps` [The prioritization and usage of attributes of meta-class `SwDataDefProps` shall follow the restrictions given in table 5.39.]()

Attributes of <code>SwDataDefProps</code>	Usage For			Place of Setting										
	RTE	A2L	Other Usage	ApplicationDataType	ImplementationDataType	DataPrototype	InstantiationDataDefProps	ParameterAccess	ComSpec	SwServiceArg	FlatInstanceDescriptor	McDataInstance	SwSystemconst	PerInstanceMemory
<code>additionalNativeTypeQualifier</code>	x		x	NA	D	I	NA	NA	NA	D	NA	I	NA	NA
<code>annotation</code>			x	D	A	A	A	A	A	D	NA	A	D	NA
<code>baseType</code>	x	x	x	NA	D	I	I	I	R	D	NA	A	M	NA
<code>compuMethod</code>	x	x	x	D	AI	I	I	NA	R	I	AI	AI	D	NA
<code>dataConstr</code>	x	x	x	D	C	R	R	I	NA	R	NA	I	D	NA
<code>displayFormat</code>		x		D	A	R	R	I	NA	R	NA	I	D	NA
<code>implementationDataType</code>	x		x	NA	D	I	I	I	NA	D	NA	NA	NA	NA
<code>invalidValue</code>	x	x		D	A	I	I	NA	D	NA	NA	I	NA	NA
<code>stepSize</code>		x		D	A	A	A	A	NA	NA	A	I	NA	NA
<code>swAddrMethod</code>	x	x	x	D	R	R	R	NA	NA	NA	R	NA	NA	D
<code>swAlignment</code>	x		x	NA	D	R	R	NA	NA	NA	NA	NA	NA	NA
<code>swBitRepresentation</code>		x	x	NA	NA	NA	NA	NA	NA	NA	NA	D	NA	NA
<code>swCalibrationAccess</code>	x	x		D	R	R	R	NA	NA	R	R	I	D	NA
<code>swCalprmAxisSet</code>	x	x		D	NA	I	I	I	NA	NA	NA	I	NA	NA
<code>swCalprmAxisSet.swCalprmAxis</code> <code>/SwAxisGrouped.swCalprmRef</code>		x		NA	NA	NA	D	R	NA	NA	NA	I	NA	NA
<code>swCalprmAxisSet.swCalprmAxis</code> <code>/SwAxisIndividual.swVariableRef</code>		x		NA	NA	NA	D	R	NA	NA	NA	I	NA	NA
<code>swCalprmAxisSet.swCalprmAxis</code> <code>/SwAxisGrouped.sharedAxisType</code>		x		D	NA	NA	NA	NA	NA	NA	NA	I	NA	NA
<code>swCalprmAxisSet.swCalprmAxis</code> <code>/SwAxisIndividual.inputVariableType</code>		x		D	NA	NA	NA	NA	NA	NA	NA	I	NA	NA
<code>swCalprmAxisSet/SwAxisIndividual.unit</code>		opt.		D	NA	I	I	I	NA	I	NA	I	NA	NA
<code>swComparisonVariable</code>		x		NA	NA	NA	NA	D	NA	NA	NA	I	NA	NA
<code>swDataDependency</code>		x	x	NA	NA	D	R	NA	NA	NA	NA	I	NA	NA
<code>swHostVariable</code>		x	x	NA	NA	NA	NA	NA	NA	NA	NA	D	NA	NA
<code>swImplPolicy</code>	x		x	D	A	A	NA	NA	NA	D	NA	NA	NA	NA
<code>swIntendedResolution</code>			x	D ¹⁰	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
<code>swInterpolationMethod</code>			x	D	I	R	R	R	NA	NA	NA	I	NA	NA
<code>swIsVirtual</code>		x		NA	NA	D	R	NA	NA	NA	NA	I	NA	NA
<code>swPointerTargetProps</code>			x	NA	D	I	NA	NA	NA	D	NA	NA	NA	NA
<code>swRecordLayout</code>	x	x	x	D	NA	I	I	I	NA	NA	NA	I	NA	NA
<code>swRefreshTiming</code>		x		D	R	R	R	NA	NA	R	NA	R	NA	NA
<code>swTextProps</code>		x	x	D	I	I	I	I	NA	NA	NA	I	NA	NA
<code>swValueBlockSize</code>		x	x	D	I	I	I	I	NA	NA	NA	I	NA	NA

¹⁰ `swIntendedResolution` is used only in an early phase of the definition of data types, namely in the context of the definition of so-called blueprints. To that extent, `swIntendedResolution` represents a non-binding requirement that shall later be considered for the definition of an appropriate `CompuMethod`.

	Usage For			Place of Setting											
Attributes of SwDataDefProps	RTE	A2L	Other Usage	ApplicationDataType	ImplementationDataType	DataPrototype	InstantiationDataDefProps	ParameterAccess	ComSpec	SwServiceArg	FlatInstanceDescriptor	McDataInstance	SwSystemconst	PerInstanceMemory	
unit		x	x	D	I	I	I	NA	NA	I	NA	I	NA	NA	
valueAxisDataType		x	x	D	I	I	I	I	NA	NA	NA	I	NA	NA	

Table 5.39: Usage of Attributes of SwDataDefProps

The following settings apply in table 5.39:

D Define the attribute independent from settings to the left.

R Use or **re-define** definition from the left in the scope of this element.

A Add attribute if not defined on the left, or as an additional information.

If the attribute has an upper multiplicity > 1 and the attribute is defined on the left then the attribute is added to the attribute defined on the left.

If the attribute has a upper multiplicity of 1 and the attribute is not defined on the left then the attribute is defined.

If the attribute has an upper multiplicity of 1 and the attribute is already defined on the left then the attribute is not redefined but this is considered as invalid configuration.

I Inherit the definition from the left for usage in the scope of this element.

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

M Attribute is **meaningless** in the scope of this element. As it was allowed in previous versions, declaring it as Not Applicable (NA) would break compatibility. Tools shall ignore such an attribute without a warning.

C This means that the left element constrains right element.

AI If the attribute is already defined on the left then the attribute is not redefined but adds implementation-related information.

Example: an `ApplicationDataType` of `category` `BOOLEAN` supports the definition of an own `CompuMethod` to define the semantics of e.g. (ON, OFF) or (HIGH, LOW) or (PASSED , FAILED) **as long as the number of values match and matching pairs of values on application level and implementation level exist**. In contrast, the corresponding `ImplementationDataType` uses (`true`, `false`) as the applicable literals in any of the above mentioned cases.

Some of the property names contain the term “variable” or “calprm”, this comes from historical¹¹ reasons and can be taken as some hint where the property most likely applies to.

Class	«atpVariation» SwDataDefProps			
Package	M2::AUTOSARTemplates::CommonStructure::DataDefProperties			
Note	This class is a collection of properties relevant for data objects under various aspects. One could consider this class as a "pattern of inheritance by aggregation". The properties can be applied to all objects of all classes in which SwDataDefProps is aggregated. Note that not all of the attributes or associated elements are useful all of the time. Hence, the process definition (e.g. expressed with an OCL or a Document Control Instance MSR-DCI) has the task of implementing limitations. SwDataDefProps covers various aspects: • Structure of the data element for calibration use cases: is it a single value, a curve, or a map, but also the recordLayouts which specify how such elements are mapped/converted to the DataTypes in the programming language (or in AUTOSAR). This is mainly expressed by properties like swRecordLayout and swCalprmAxisSet • Implementation aspects, mainly expressed by swImplPolicy, swVariableAccessImplPolicy, swAddrMethod, swPointerTagetProps, baseType, implementationDataType and additionalNativeTypeQualifier • Access policy for the MCD system, mainly expressed by swCalibrationAccess • Semantics of the data element, mainly expressed by compuMethod and/or unit, dataConstr, invalidValue • Code generation policy provided by swRecordLayout Tags: vh.latestBindingTime=codeGenerationTime			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
additionalNativeTypeQualifier	NativeDeclarationString	0..1	attr	This attribute is used to declare native qualifiers of the programming language which can neither be deduced from the baseType (e.g. because the data object describes a pointer) nor from other more abstract attributes. Examples are qualifiers like "volatile", "strict" or "enum" of the C-language. All such declarations have to be put into one string. Tags: xml.sequenceOffset=235

¹¹In the beginning of ASAM and MSR measurements and calibration parameters (characteristics) were separated and the properties were merged over the time.

Attribute	Datatype	Mul.	Kind	Note
annotation	Annotation	*	aggr	This aggregation allows to add annotations (yellow pads ...) related to the current data object. Tags: xml.roleElement=true; xml.roleWrapperElement=true; xml.sequenceOffset=20; xml.typeElement=false; xml.typeWrapperElement=false
baseType	SwBaseType	0..1	ref	Base type associated with the containing data object. Tags: xml.sequenceOffset=50
compuMethod	CompuMethod	0..1	ref	Computation method associated with the semantics of this data object. Tags: xml.sequenceOffset=180
dataConstr	DataConstr	0..1	ref	Data constraint for this data object. Tags: xml.sequenceOffset=190
displayFormat	DisplayFormatString	0..1	attr	This property describes how a number is to be rendered e.g. in documents or in a measurement and calibration system. Tags: xml.sequenceOffset=210
implementationDataType	ImplementationDataType	0..1	ref	This association denotes the ImplementationDataType of a data declaration via its aggregated SwDataDefProps. It is used whenever a data declaration is not directly referring to a base type. Especially • redefinition of an ImplementationDataType via a "typedef" to another ImplementationDatatype • the target type of a pointer (see SwPointerTargetProps), if it does not refer to a base type directly • the data type of an array or record element within an ImplementationDataType, if it does not refer to a base type directly • the data type of an SwServiceArg, if it does not refer to a base type directly Tags: xml.sequenceOffset=215
invalidValue	ValueSpecification	0..1	aggr	Optional value to express invalidity of the actual data element. Tags: xml.sequenceOffset=255
stepSize	Float	0..1	attr	This attribute can be used to define a value which is added to or subtracted from the value of a DataPrototype when using up/down keys while calibrating.

Attribute	Datatype	Mul.	Kind	Note
swAddrMethod	SwAddrMethod	0..1	ref	Addressing method related to this data object. Via an association to the same SwAddrMethod it can be specified that several DataPrototypes shall be located in the same memory without already specifying the memory section itself. Tags: xml.sequenceOffset=30
swAlignment	AlignmentType	0..1	attr	The attribute describes the intended alignment of the DataPrototype. If the attribute is not defined the alignment is determined by the swBaseType size and the memoryAllocationKeywordPolicy of the referenced SwAddrMethod. Tags: xml.sequenceOffset=33
swBitRepresentation	SwBitRepresentation	0..1	aggr	Description of the binary representation in case of a bit variable. Tags: xml.sequenceOffset=60
swCalibrationAccess	SwCalibrationAccessEnum	0..1	attr	Specifies the read or write access by MCD tools for this data object. Tags: xml.sequenceOffset=70
swCalprmAxisSet	SwCalprmAxisSet	0..1	aggr	This specifies the properties of the axes in case of a curve or map etc. This is mainly applicable to calibration parameters. Tags: xml.sequenceOffset=90
swComparisonVariable	SwVariableRefProxy	*	aggr	Variables used for comparison in an MCD process. Tags: xml.sequenceOffset=170; xml.type Element=false
swDataDependency	SwDataDependency	0..1	aggr	Describes how the value of the data object has to be calculated from the value of another data object (by the MCD system). Tags: xml.sequenceOffset=200
swHostVariable	SwVariableRefProxy	0..1	aggr	Contains a reference to a variable which serves as a host-variable for a bit variable. Only applicable to bit objects. Tags: xml.sequenceOffset=220; xml.type Element=false
swImplPolicy	SwImplPolicyEnum	0..1	attr	Implementation policy for this data object. Tags: xml.sequenceOffset=230

Attribute	Datatype	Mul.	Kind	Note
swIntendedResolution	Numerical	0..1	attr	The purpose of this element is to describe the requested quantization of data objects early on in the design process. The resolution ultimately occurs via the conversion formula present (compuMethod), which specifies the transition from the physical world to the standardized world (and vice-versa) (here, "the slope per bit" is present implicitly in the conversion formula). In the case of a development phase without a fixed conversion formula, a pre-specification can occur through swIntendedResolution. The resolution is specified in the physical domain according to the property "unit". Tags: xml.sequenceOffset=240
swInterpolationMethod	Identifier	0..1	ref	This is a keyword identifying the mathematical method to be applied for interpolation. The keyword needs to be related to the interpolation routine which needs to be invoked. Tags: xml.sequenceOffset=250
swIsVirtual	Boolean	0..1	attr	This element distinguishes virtual objects. Virtual objects do not appear in the memory, their derivation is much more dependent on other objects and hence they shall have a swDataDependency . Tags: xml.sequenceOffset=260
swPointerTargetProps	SwPointerTargetProps	0..1	aggr	Specifies that the containing data object is a pointer to another data object. Tags: xml.sequenceOffset=280
swRecordLayout	SwRecordLayout	0..1	ref	Record layout for this data object. Tags: xml.sequenceOffset=290
swRefreshTiming	MultidimensionalTime	0..1	aggr	This element specifies the frequency in which the object involved shall be or is called or calculated. This timing can be collected from the task in which write access processes to the variable run. But this cannot be done by the MCD system. So this attribute can be used in an early phase to express the desired refresh timing and later on to specify the real refresh timing. Tags: xml.sequenceOffset=300

Attribute	Datatype	Mul.	Kind	Note
swTextProps	SwTextProps	0..1	aggr	the specific properties if the data object is a text object. Tags: xml.sequenceOffset=120
swValueBlockSize	Numerical	0..1	attr	This represents the size of a Value Block Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime xml.sequenceOffset=80
unit	Unit	0..1	ref	Physical unit associated with the semantics of this data object. This attribute applies if no compuMethod is specified. If both units (this as well as via compuMethod) are specified the units shall be compatible. Tags: xml.sequenceOffset=350
valueAxisDataType	ApplicationPrimitiveDataType	0..1	ref	The referenced ApplicationPrimitiveDataType represents the primitive data type of the value axis within a compound primitive (e.g. curve, map). It supersedes CompuMethod, Unit, and BaseType. Tags: xml.sequenceOffset=355

Table 5.40: SwDataDefProps

Primitive	NativeDeclarationString
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::PrimitiveTypes
Note	This string contains a native data declaration of a data type in a programming language. It is basically a string, but white-space must be preserved. Tags: xml.xsd.customType=NATIVE-DECLARATION-STRING; xml.xsd.type=string; xml.xsd.whiteSpace=preserve

Table 5.41: NativeDeclarationString

Class	SwBitRepresentation			
Package	M2::AUTOSARTemplates::CommonStructure::DataDefProperties			
Note	Description of the structure of a bit variable: Comprises of the bitPosition in a memory object (e.g. swHostVariable, which stands parallel to swBitRepresentation) and the numberOfBits . In this way, interrelated memory areas can be described. Non-related memory areas are not supported.			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note

Attribute	Datatype	Mul.	Kind	Note
bitPosition	Integer	0..1	attr	If the "bit data object" is hosted within another data object (e.g. if the memory can be accessed via byte as well as bit address), this attribute specifies the position of the data object. The count starts at zero (0). Tags: xml.sequenceOffset=20
numberOfBits	Integer	0..1	attr	Number of bits allocated by a "bit data object" within its host data object. Tags: xml.sequenceOffset=30

Table 5.42: SwBitRepresentation

Primitive	DisplayFormatString
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::PrimitiveTypes

Note	This is a display format specifier for the display of values e.g. in documents or in measurement and calibration systems. The display format specifier is a subset of the ANSI C printf specifiers with the following form: <pre>% [flags] [width] [.prec] type character</pre> For more details refer to "ASAM-HarmonizedDataObjects-V1.1.pdf" chapter 13.3.2 DISPLAY OF DATA. Due to the numerical nature of value settings, only the following type characters are allowed: • d: Signed decimal integer • i: Signed decimal integer • o: Unsigned octal integer • u: Unsigned decimal integer • x: Unsigned hexadecimal integer, using "abcdef" • X: Unsigned hexadecimal integer, using "ABCDEF" • e: Signed value having the form [-]d.dddd e [sign]ddd where d is a single decimal digit, dddd is one or more decimal digits, ddd is exactly three decimal digits, and sign is + or - • E: Identical to the e format except that E rather than e introduces the exponent • f: Signed value having the form [-]dddd.dddd, where dddd is one or more decimal digits; the number of digits before the decimal point depends on the magnitude of the number, and the number of digits after the decimal point depends on the requested precision • g: Signed value printed in f or e format, whichever is more compact for the given value and precision; trailing zeros are truncated, and the decimal point appears only if one or more digits follow it • G: Identical to the g format, except that E, rather than e, introduces the exponent (where appropriate) Tags: xml.xsd.customType=DISPLAY-FORMAT-STRING; xml.xsd.pattern=%[\-+#]?[0-9]*(\.[0-9])?[diouxXfeEgGcs]; xml.xsd.type=string

Table 5.43: DisplayFormatString

Class	Annotation			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::Documentation::Annotation			
Note	This is a plain annotation which does not have further formal data.			
Base	ARObject, GeneralAnnotation			
Attribute	Datatype	Mul.	Kind	Note
—	—	—	—	—

Attribute	Datatype	Mul.	Kind	Note

Table 5.44: Annotation

[constr_1244] DataPrototypes used in application software shall not be typed by C enums [A [DataPrototype](#) that is used in an [AtomicSwComponentType](#) shall not set [swDataDefProps.additionalNativeTypeQualifier](#) to enum.]()

[TPS_SWCT_01272] Semantics of [swComparisonVariable](#) [Please note that [swComparisonVariables](#) shall be displayed in the MCD system on the ordinate in a curve. By showing the input value and the comparison value the calibration engineer can see if the current working point is above or below a curve provident thresholds. For example in a curve specifying a temperature depending gear shift threshold engine speed the engine speed can be shown as “comparisonVariable”.

These variables can be used to display the value of a variable on the value axis of a calibration parameter (characteristic), that is currently displayed in the MCD-System. The purpose is to compare the appropriate result from the calibration parameter in question, with a value being calculated or taken from a sensor (the comparison variable).

The sole purpose of this comparison-variable is therefore to serve the calibration process.]()

The meaning behind [swComparisonVariable](#) is depicted in Figure 5.24. Legend: t_x represents the current temperature and t_{mot} represents the motor temperature. V represents the current speed as shown in the MCD system for comparison: this is the [swComparisonVariable](#). Likewise, V_s represents the speed characteristic over the temperature.

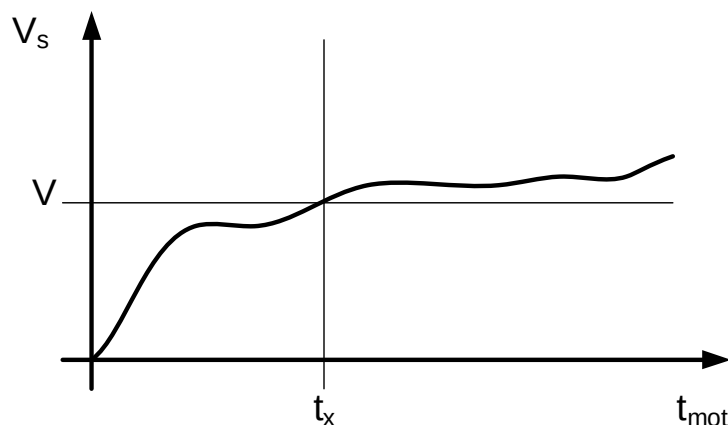


Figure 5.24: Explanation of [swComparisonVariable](#)

Enumeration	SwCalibrationAccessEnum
Package	M2::AUTOSARTemplates::CommonStructure::DataDefProperties
Note	Determines the access rights to a data object w.r.t. measurement and calibration.
Literal	Description
notAccessi- ble	The element will not be accessible via MCD tools, i.e. will not appear in the ASAP file.

readOnly	The element will only appear as read-only in an ASAP file.
readWrite	The element will appear in the ASAP file with both read and write access.

Table 5.45: SwCalibrationAccessEnum

[TPS_SWCT_01273] Precedence rules for the application of [SwDataDefProps](#) [[SwDataDefProps](#) can be specified on various levels, from type over prototype to instantiation, finally data access and calibration support after RTE generation. In general, properties specified on prototype level override the ones specified on type level.

More formally, the precedence of such properties is:

1. attributes of [SwDataDefProps](#) defined on [ApplicationDataType](#) which may be overwritten by
2. attributes of [SwDataDefProps](#) defined on [ImplementationDataType](#) which may be overwritten by
3. attributes of [SwDataDefProps](#) defined on [DataPrototype](#) which may be overwritten by
4. attributes of [SwDataDefProps](#) defined on [InstantiationDataDefProps](#) which may be overwritten by
5. attributes of [SwDataDefProps](#) defined on [ParameterAccess](#) respectively [Argument](#) which may be overwritten by
6. attributes of [SwDataDefProps](#) defined on [FlatInstanceDescriptor](#) which may be overwritten by
7. attributes of [SwDataDefProps](#) defined on [McDataInstance](#)

]()

Note that details about applicable attributes of [SwDataDefProps](#) can be found in Table 5.39.

[TPS_SWCT_01274] [SwDataDefProps](#) used to support calibration and measurement [The last item in the list of use cases contained in [TPS_SWCT_01273] denotes that [SwDataDefProps](#) are also used as part of [McSupportData](#) which is a direct input to the generation of measurement and calibration configuration formats (so-called A2L-files). This use case is further explained in [7]. Since these data are generated by the RTE, they will use a copy of the properties according to the precedence given above.

However, even in this use case which comes after RTE generation it is possible that properties relevant for the MCD system are added which had been undefined so far.

This for example, applies to the attribute [swRefreshTiming](#) which denotes a timing information relevant for the measurement system; this information may be set rather late in the process chain.]()

Obviously such an override is not applicable in all cases. In particular, the properties covering the structure shall not be redefined on [DataPrototype](#). Implementation policy, semantics and code generation policy may be changed under consideration of compatibility rules.

Access policy for the MCD system is the most likely subject to be redefined on the [DataPrototype](#) of even on an instantiation level.

Section 5.4.3 describes how [SwDataDefProps](#) are used for measuring purposes while Section 5.4.4 describes the construction of characteristics based on the combination of [SwDataDefProps](#) with [DataPrototypes](#).

Section 2.2.2 describes in which context calibration parameters can be defined. Finally, sections 2.2.3, 7.5.4, and 5.5.4 show how calibration parameters are used in [RunnableEntities](#) and show the link to an actual ECU implementation.

Enumeration	SwImplPolicyEnum
Package	M2::AUTOSARTemplates::CommonStructure::DataDefProperties
Note	Specifies the implementation strategy with respect to consistency mechanisms of variables.
Literal	Description
const	forced implementation such that the running software within the ECU shall not modify it. For example implemented with the "const" modifier in C. This can be applied for parameters (not for those in NVRAM) as well as argument data prototypes.
fixed	This data element is fixed. In particular this indicates, that it might also be implemented e.g. as in place data, (#DEFINE).
measurement Point	The data element is created for measurement purposes only. The data element is never read directly within the ECU software. In contrast to a "standard" data element in an unconnected provide port is, this unconnection is guaranteed for measurementPoint data elements.
queued	The content of the data element is queued and the data element has 'event' semantics, i.e. data elements are stored in a queue and all data elements are processed in 'first in first out' order. The queuing is intended to be implemented by RTE Generator. This value is not applicable for parameters.
standard	This is applicable for all kinds of data elements. For variable data prototypes the 'last is best' semantics applies. For parameter there is no specific implementation directive.

Table 5.46: SwImplPolicyEnum

[TPS_SWCT_01275] values of the attribute [swImplPolicy](#) are restricted depending on the context [The values of the attribute [swImplPolicy](#) are restricted depending on the context. This restriction reflects the fact that not all possible implementation strategies are useful or supported for all kinds of [DataPrototypes](#).]()

These restrictions are summarized in table 5.47 and formalized in the following constraints. Please note that the usage of [swImplPolicy](#) is further constraint in the combination with the attribute value [swCalibrationAccess](#) as described in [[constr_1017](#)].

Attribute of <code>SwImplPolicyEnum</code>	VariableDataPrototype							ParameterDataPrototype					Misc.	
	VariableDataPrototype in SenderReceiverInterface	VariableDataPrototype in NvDataInterface	VariableDataPrototype in role ramBlock	VariableDataPrototype in role implicitInterRunnableVariable	VariableDataPrototype in role explicitInterRunnableVariable	VariableDataPrototype in role arTypedPerInstanceMemory	VariableDataPrototype in role staticMemory	ParameterDataPrototype in ParameterInterface	ParameterDataPrototype in role romBlock	ParameterDataPrototype in role sharedParameter	ParameterDataPrototype in role perInstanceParameter	ParameterDataPrototype in role constantMemory	ArgumentDataPrototype	SwServiceArg
<code>const</code>	NA	NA	NA	NA	NA	NA	NA	x	NA	NA	NA	x	NA	x
<code>fixed</code>	NA	NA	NA	NA	NA	NA	NA	x	NA	NA	NA	x	NA	NA
<code>measurementPoint</code>	x	NA	NA	NA	NA	x	x	NA	NA	NA	NA	NA	NA	NA
<code>queued</code>	x	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
<code>standard</code>	x	x	x	x	x	x	x	x	x	x	x	x	x	x
<code>message</code>	NA	NA	NA	NA	NA	NA	x	NA	NA	NA	NA	NA	NA	NA

Table 5.47: Allowed attributes values for `SwImplPolicy` vs. `DataPrototypes` and their roles

The following settings apply in table 5.47:

x Attribute is applicable for usage in the scope of this element.

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

[constr_2035] `swImplPolicy` for `VariableDataPrototype` in `SenderReceiverInterface` [The overriding `swImplPolicy` attribute value of a `VariableDataPrototype` in `SenderReceiverInterface` shall be `standard`, `queued` or `measurementPoint`.]()

[constr_2036] `swImplPolicy` for `VariableDataPrototype` in `NvDataInterface` [The overriding `swImplPolicy` attribute value of a `VariableDataPrototype` in `NvDataInterface` shall be `standard`.]()

[constr_2037] `swImplPolicy` for `VariableDataPrototype` in the role `ramBlock` [The overriding `swImplPolicy` attribute value of a `VariableDataPrototype` in the role `ramBlock` shall be `standard`.]()

[constr_2038] `swImplPolicy` for `VariableDataPrototype` in the role `implicitInterRunnableVariable` [The overriding `swImplPolicy` attribute value of a `VariableDataPrototype` in the role `implicitInterRunnableVariable` shall be `standard`.]()

[constr_2039] `swImplPolicy` for `VariableDataPrototype` in the role `explicitInterRunnableVariable` [The overriding `swImplPolicy` attribute value of a `VariableDataPrototype` in the role `explicitInterRunnableVariable` shall be `standard`.]()

[constr_2040] `swImplPolicy` for `VariableDataPrototype` in the role `arTypedPerInstanceMemory` [The overriding `swImplPolicy` attribute value of a `VariableDataPrototype` in the role `arTypedPerInstanceMemory` shall be `standard` or `measurementPoint`.]()

[constr_2041] `swImplPolicy` for `VariableDataPrototype` in the role `staticMemory` [The overriding `swImplPolicy` attribute value of a `VariableDataPrototype` in the role `staticMemory` shall be `standard`, `measurementPoint` or `message`.]()

[constr_2042] `swImplPolicy` for `ParameterDataPrototype` in `ParameterInterface` [The overriding `swImplPolicy` attribute value of a `ParameterDataPrototype` in `ParameterInterface` shall be `standard`, `const` or `fixed`.]()

[constr_2043] `swImplPolicy` for `ParameterDataPrototype` in the role `staticMemory` [The overriding `swImplPolicy` attribute value of a `ParameterDataPrototype` in the role `romBlock` shall be `standard`.]()

[constr_2044] `swImplPolicy` for `ParameterDataPrototype` in the role `sharedParameter` [The overriding `swImplPolicy` attribute value of a `ParameterDataPrototype` in the role `sharedParameter` shall be `standard`.]()

[constr_2045] `swImplPolicy` for `ParameterDataPrototype` in the role `perInstanceParameter` [The overriding `swImplPolicy` attribute value of a `ParameterDataPrototype` in the role `sharedParameter` shall be `standard`.]()

[constr_2046] `swImplPolicy` for `ParameterDataPrototype` in the role `constantMemory` [The overriding `swImplPolicy` attribute value of a `ParameterDataPrototype` in the role `sharedParameter` shall be `standard`, `const` or `fixed`.]()

[constr_2047] `swImplPolicy` for `ArgumentDataPrototype` [The overriding `swImplPolicy` attribute value of a `ArgumentDataPrototype` shall be `standard`.]()

[constr_2048] `swImplPolicy` for `SwServiceArg` [The overriding `swImplPolicy` attribute value of a `SwServiceArg` shall be `standard` or `const`.]()

[TPS_SWCT_02000] Default value for attribute `swImplPolicy` [If the attribute `swImplPolicy` is not explicitly set at any of the locations listed in "Place of Setting"

for `SwDataDefProps` mentioned in table 5.39 the default value `standard` applies.
>()

5.4.2 Invalid Value

The diagram 5.5 shows that in addition to the semantics defined through the `compuMethod` (explained below in chapter 5.5.1), also an `invalidValue` can be specified. This is a requirement of the VFB [3], allowing to express which specific value is used to indicate invalidation.

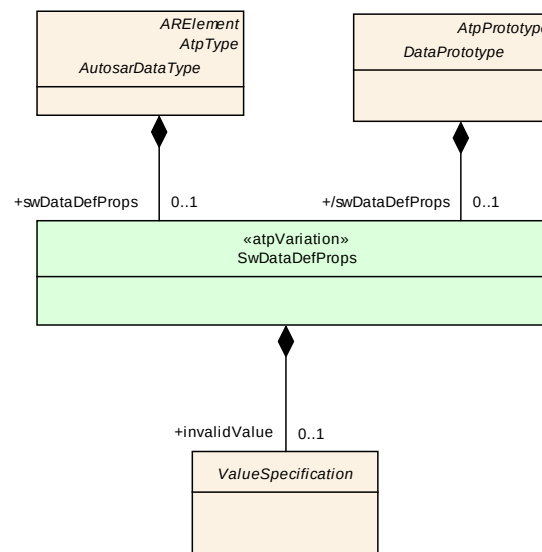


Figure 5.25: Invalid value

The `invalidValue` can be used in different flavors (also illustrated in Figure 5.6:

- **[TPS_SWCT_01432] Keep the `invalidValue` transparent to the sending and receiving software components** [On the one hand it is possible to keep the `invalidValue` transparent to the sending and receiving software components. In this case the invalidation API of the RTE on the sender side has to be used.

The receiving software component can either use the data receive status or the `DataReceiveErrorEvent` respectively `DataReceivedEvent` to decide about the validity of the received data or the receiving software component can rely on the reception of an `initValue` as a default value in case of data invalidation.

In this case the invalid value should (and usually will) be **outside of the range limits** defined by the `compuMethod`.]()

- **[TPS_SWCT_01434] Sender and receiver have knowledge of invalid value** [On the other hand it is possible that the communicating software components do have knowledge about the `invalidValue` and the `invalidValue` is visible for them.

This is in particular the case if the sender and receiver are calculating a checksum over a larger data structure to implement an end to end communication protection. To ensure the integrity of the checksums it is required to set invalid values by the sending component directly and to receive invalid values unchanged.

In this case the invalid value should (and usually will) be **inside of the range limits** defined by the `compuMethod`. `]()`

- **[TPS_SWCT_01436] Different receivers require different handling of data invalidation** `]()` It is possible that in case of 1:n communication different receivers requiring a different handling of data invalidation depending on the criticality of its functionality. For instance, one receiver applies the checksum based end to end communication protection and another receiver relies on the substitution of invalid values by `invalidValues`. `]()`

A typical use case for putting the `invalidValue` inside the boundaries of the applicable `CompuMethod` is a composite data type that contains the values of all individual wheel speeds. If one of the sensors fails and starts to send `invalidValue` it would probably not make sense to consider the whole composite data element invalid.

It may very likely still be possible to make sense of the remaining intact wheel speed values and carry on with whatever business the receiving software-component has with that data.

From this perspective, it would obviously be OK for the sending software-component to actively send the `invalidValue` that is then processed as a “regular” value without applying additional semantics by the RTE/Com.

[TPS_SWCT_01646] Sending `invalidValue` without invalidation applied by RTE/Com `]()` For intentionally sending `invalidValue` without invalidation applied by RTE/Com the `SenderReceiverInterface.invalidationPolicy.handleInvalid` shall be set to the value `HandleInvalidEnum.dontInvalidate`. `]()`

[constr_1390] Restriction to the value of `SenderReceiverInterface.invalidationPolicy.handleInvalid` `]()` If the value of `SenderReceiverInterface.invalidationPolicy.handleInvalid` is set to any value other than `HandleInvalidEnum.dontInvalidate` then the `invalidValue` shall not be within the interval defined by the `CompuMethod` of the applicable `dataElement`. `]()`

Please note that `ApplicationPrimitiveDataTypes` of category `VALUE` in principle can have an `invalidValue` provided by a `NumericalValueSpecification` because the value of the attribute `invalidValue` can be **outside the range** of the applicable `CompuMethod` (see [\[TPS_SWCT_01432\]](#)).

[TPS_SWCT_01437] `invalidValue` can also be specified without setting a `compuMethod` `]()` An `invalidValue` can also be specified without setting a `compuMethod`. `]()`

Figure 5.6 illustrates the relationship between `ApplicationDataType`, `CompuMethod`, `ImplementationDataType`, `invalidValue`, `BaseType`.

[constr_2545] `invalidValue` shall fit in the specified ranges [The `invalidValue` shall be in the range of the `ImplementationDataType`.]()

Please note that the `invalidValue` is a `ValueSpecification`. Of course, it would technically be possible to use any subclass of `ValueSpecification` at this place.

[constr_1016] Restriction of `invalidValue` for `ImplementationDataType` and `ImplementationDataTypeElement` [`invalidValue` for `ImplementationDataType` and `ImplementationDataTypeElement` is restricted to to be either a compatible `NumericalValueSpecification`, `TextValueSpecification` (caution, [constr_1284] applies) or a `ConstantReference` that in turn points to a compatible `ValueSpecification`.]()

[constr_1384] Definition of `invalidValue` for `DataPrototype` typed by `ApplicationPrimitiveDataType` of category `CURVE`, `MAP`, `CUBOID`, `CUBE_4`, `CUBE_5`, `COM_AXIS`, `RES_AXIS`, and `VAL_BLK` [An `invalidValue` shall not be specified for a `DataPrototype` typed by `ApplicationPrimitiveDataType` of category `CURVE`, `MAP`, `CUBOID`, `CUBE_4`, `CUBE_5`, `COM_AXIS`, `RES_AXIS`, and `VAL_BLK`]()

Rationale for [constr_1384]: there is no use case for sending a `DataPrototype` typed by `ApplicationPrimitiveDataType` of category `CURVE`, `MAP`, `CUBOID`, `CUBE_4`, `CUBE_5`, `COM_AXIS`, `RES_AXIS`, and `VAL_BLK` over a communication bus.

[constr_1242] Restriction of `invalidValue` for `ApplicationPrimitiveDataType` of category `STRING` [`invalidValue` for `ApplicationPrimitiveDataType` of category `STRING` ([constr_1241] applies) is restricted to be either a compatible `ApplicationValueSpecification` or a `ConstantReference` that in turn points to a compatible `ApplicationValueSpecification`.]()

[TPS_SWCT_01487] Correspondence of `invalidValue` for `ApplicationPrimitiveDataType` and `ImplementationDataType` [The `invalidValue` specified on the level of an `ApplicationPrimitiveDataType` shall correspond to the `invalidValue` specified on the level of a compatible `ImplementationDataType`. The terms “corresponds” boils down to:

- category `VALUE` or `BOOLEAN`: application of `CompuMethod`
- category `STRING`: mapping of the encoding on the `ApplicationPrimitiveDataType` side to the numerical values on the level of the `ImplementationDataType` (shall reference `SwBaseType` with `baseTypeEncoding` set to `NONE`). There is no formal support defined to check that the values of `invalidValue` really correspond to each other.

]()

[constr_1225] `DataPrototype` is typed by an `ImplementationDataType` that references a `CompuMethod` of category `TEXTTABLE` or `BITFIELD_TEXTTABLE` [If a `DataPrototype` is typed by an `ImplementationDataType` that references a `CompuMethod` of category `TEXTTABLE` or `BITFIELD_TEXTTABLE` the applicable `ValueSpecification` shall be a `TextValueSpecification`. In this case the

value provided shall match to one of the applicable text values (`vt`, `shortLabel`, `symbol`) defined by the applicable `CompuScales`. `]()`

[TPS_SWCT_01467] `ImplementationDataType` references an `SwBaseType` with a string encoding `[` If an `ImplementationDataType` references an `SwBaseType` with a string encoding the `initValue` shall still be provided as numerical values according to the string encoding. `]()`

[constr_1302] Restriction of data invalidation `[` Data invalidation is only applicable for one of the following cases applicable on the **receiving** side:

1. `VariableDataPrototypes` typed by either an `ApplicationPrimitiveDataType` or an `ImplementationDataType` of category `VALUE` or `TYPE_REFERENCE` that boils down to category `VALUE` that have defined an `invalidValue`.
2. `VariableDataPrototypes` typed by either an `ApplicationCompositeDataType` or an `ImplementationDataType` of category `STRUCTURE`, or `ARRAY` or of category `TYPE_REFERENCE` that boils down to category `STRUCTURE`, or `ARRAY` that have **at least one** primitive element with an `invalidValue`.
3. `VariableDataPrototypes` typed by an `ImplementationDataType` of category `UNION` or of category `TYPE_REFERENCE` that boils down to category `UNION` where **all** primitive elements define an `invalidValue`.

`]()`

[constr_1140] Combination of `invalidValue` with the attribute `handleInvalid` `[` The combination of setting the attribute `handleInvalid` of the meta-class `InvalidationPolicy` owned by `SenderReceiverInterface` to value `replace` **and** of setting the value of the attribute `initValue` owned by a corresponding `NonqueuedReceiverComSpec` effectively to the value of the `invalidValue` (owned by a corresponding `SwDataDefProps`) is not supported. `]()`

The term “corresponding” (as utilized in [constr_1140]) refers to the fact that information regarding the fulfillment of [constr_1140] is factually distributed over different areas of the meta-model. For clarification, the following relationship should be considered:

The `SenderReceiverInterface` defines how to deal with an invalid value by means of the attribute `handleInvalid` on the basis of individual `dataElements`. The `SenderReceiverInterface` is taken for typing a `RPortPrototype` that in turn owns a `ReceiverComSpec`. [constr_1140] applies if the particular `ReceiverComSpec` is actually a `NonqueuedReceiverComSpec` that refers to the same `dataElement`.

In this case the `invalidValue` owned by the `SwDataDefProps` that in turn is owned by the respective `dataElement` is relevant for the fulfillment of [constr_1140]. The “big picture” of this relationship is sketched in Figure 5.26.

[constr_1219] Invalidation depends on the value of **swImplPolicy** [Invalidation of **dataElements** is only supported for **dataElements** where the value of **swImplPolicy** is **not** set to **queued**.]()

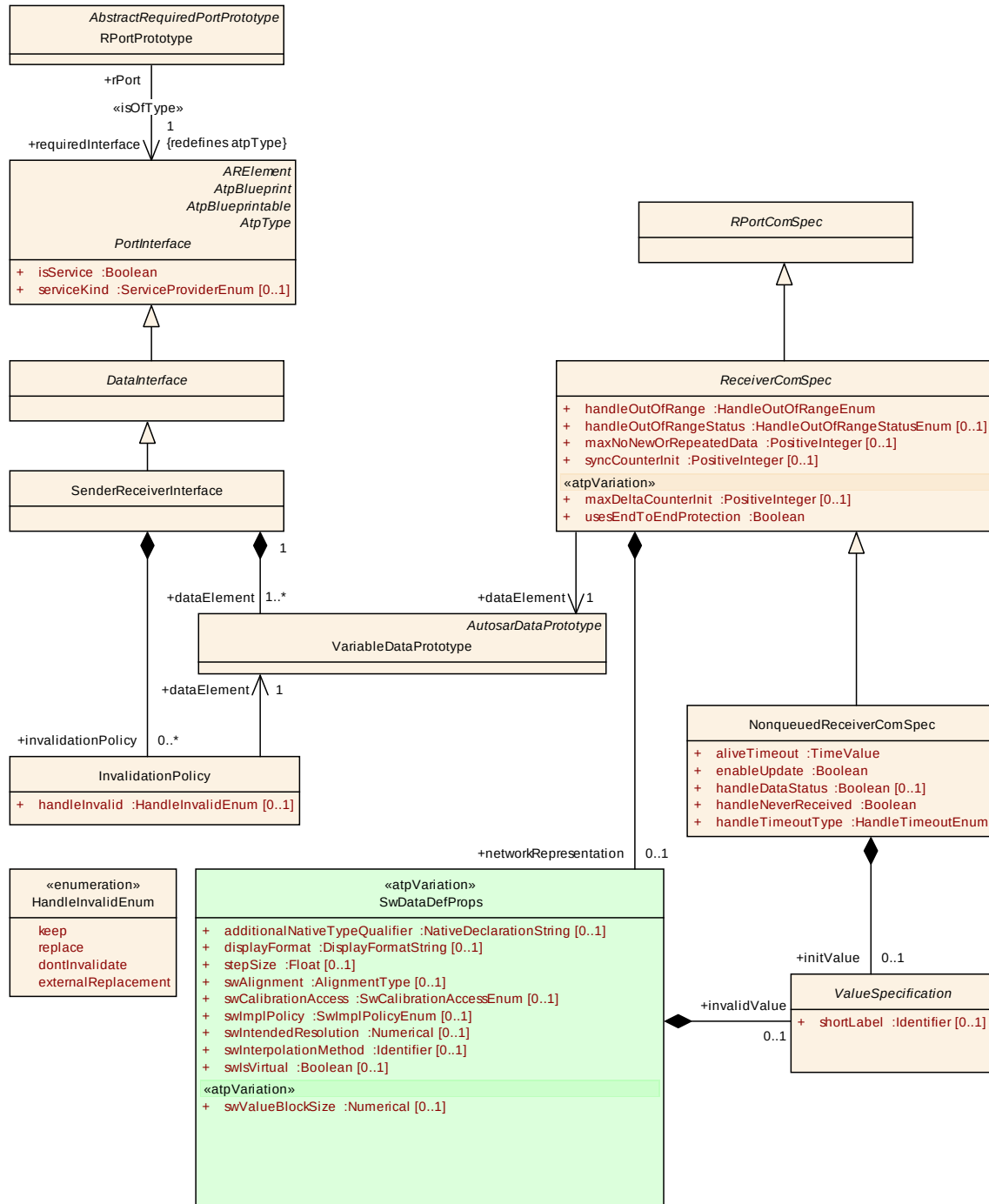


Figure 5.26: Relationships required to consider the **invalidValue**

[constr_1282] Restriction concerning the usage of **RuleBasedValueSpecification** or a **ReferenceValueSpecification** for the specification of an **invalidValue** [The aggregation of a **RuleBasedValueSpecification** or a **Ref-**

erenceValueSpecification for the definition of a `ApplicationPrimitive-DataType.swDataDefProps.invalidValue` is not supported. `]()`

5.4.3 Properties for Measurement

In embedded automotive software design, measurement means access to memory locations in an ECU and transferring its contents to the measurement & calibration system. While in classical software design, variables abstract the memory locations in the code, AUTOSAR provides for this purpose the `DataPrototype` with its various specializations:

- `VariableDataPrototype` of a `SenderReceiverInterface` or `NvDataInterface` used in a `PortPrototype` (of a `SwComponentPrototype`), to capture sender-receiver and non volatile data communication between `SwComponentPrototypes`
- `ArgumentDataPrototype` of a `ClientServerOperation` in a `ClientServerInterface` to capture client-server communication between `SwComponentPrototypes`.
- `VariableDataPrototype` in the context of an `SwcInternalBehavior` to
 - capture communication between `RunnableEntities` within a `SwComponentPrototype`
 - handle data in a non volatile memory block
 - provide pure software component internal memory which has to be accessible for a MCD system

[TPS_SWCT_01440] Measurement is not limited to primitive objects [The ability of being measured is not restricted to primitive data (`category VALUE`) but can also be applied to composite data (`category STRUCTURE` or `ARRAY`). `]()`

The following semantical and structural features from `SwDataDefProps` are relevant (among other purposes) for the measurement system:

- `swCalibrationAccess`
- `swImplPolicy`
- `compuMethod`
- `unit` (if not specified by `compuMethod`)
- `baseType`
- `swAddrMethod`

[TPS_SWCT_01130] Measurement and calibration access to model elements is defined by `swCalibrationAccess` [The ability to be accessed by e.g. a calibration tool is given by setting the `swCalibrationAccess` attribute.] ([RS_SWCT_03152](#))

The following table shows all valid settings of `swCalibrationAccess`:

Enumeration	SwCalibrationAccessEnum
Package	M2::AUTOSARTemplates::CommonStructure::DataDefProperties
Note	Determines the access rights to a data object w.r.t. measurement and calibration.
Literal	Description
notAccessible	The element will not be accessible via MCD tools, i.e. will not appear in the ASAP file.
readOnly	The element will only appear as read-only in an ASAP file.
readWrite	The element will appear in the ASAP file with both read and write access.

Table 5.48: SwCalibrationAccessEnum

[TPS_SWCT_01559] Default value for attribute `SwDataDefProps.swCalibrationAccess` [The default value for the attribute `SwDataDefProps.swCalibrationAccess` is `SwCalibrationAccessEnum.notAccessible`.]()

[constr_1017] Supported combinations of `swImplPolicy` and `swCalibrationAccess` [The table 5.49 defines the supported combinations of `swImplPolicy` and `swCalibrationAccess` attribute setting.]()

<code>swImplPolicy</code>	<code>swCalibrationAccess</code>		
	<code>notAccessible</code>	<code>readOnly</code>	<code>readWrite</code>
<code>fixed</code>	yes	not supported	not supported
<code>const</code>	yes	yes	not supported
<code>standard</code>	yes	yes	yes
<code>queued</code>	yes	not supported	not supported
<code>measurementPoint</code>	not supported	yes	not supported

Table 5.49: Supported combinations of `swImplPolicy` and `swCalibrationAccess`

[constr_1018] `measurementPoint` shall not be referenced by a `VariableAccess` aggregated by `RunnableEntity` in the role `dataReadAccess` [Due to the nature of `data elements` characterized by setting the `swImplPolicy` to `measurementPoint`, such `data elements` shall not be referenced by a `VariableAccess` aggregated by `RunnableEntity` in the role `dataReadAccess`.]()

5.4.4 Properties of Curves and Maps

A characteristic table is defined by setting the `category` of the corresponding `AutosarDataType` or `DataPrototype` to `CURVE` respectively `MAP`, `CUBOID`, `CUBE_4`, and `CUBE_5`.

Its `SwDataDefProps` determine an axis description. The type of the functional values is given by the attached `SwBaseType` and the `CompuMethod`.

The axis description itself is defined by the meta-model element `SwCalprmAxisSet` aggregating the appropriate number of `SwCalprmAxisTypeProps`.

This is the base class for a so called “individual axis” (formalized by meta-class `SwAxisIndividual`) or a “grouped axis” (formalized by meta-class `SwAxisGrouped`).

The latter is used to share axis points by several characteristic tables. Figure 5.27 shows an overview on the relevant meta-model elements.

The type of the functional values is given by the attached `SwBaseType` and the `CompuMethod` or by the referenced `ApplicationDataType`.

If an `ApplicationDataType` is referenced (via `valueAxisDataType`) this supersedes `CompuMethod`, `Unit`, and `BaseType` if these are defined in parallel.

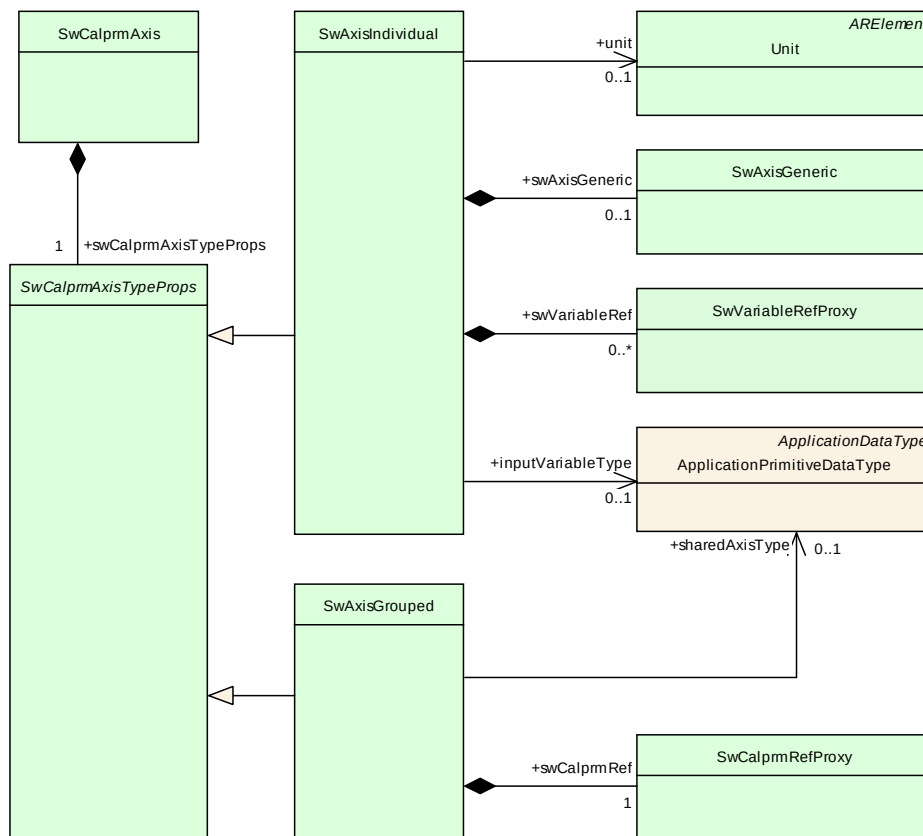


Figure 5.27: Overview on the Meta-Model for Axis Description

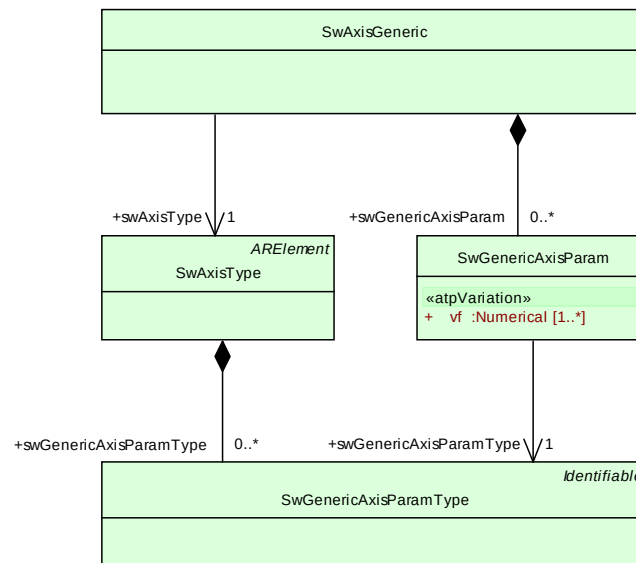


Figure 5.28: Overview on a Generic Axis

Figure 5.29 shows how an individual axis is represented by the meta-model. The corresponding M1 Model is illustrated in Figure 5.30. The `SwAxisIndividual` references value-models to account the minimum and the maximum number of axis values as well as the number of axis points.

Hence, the size of the structure to hold the functional values is determined by the number of axis values for all axes. The type of the axis values is determined when the type of the referenced input value (`swVariableRef`) has been set. For further details see 5.4.5.

[TPS_SWCT_01107] `swMinAxisPoints` and `swMaxAxisPoints` represent variation points [The value of attributes `swMinAxisPoints` and `swMaxAxisPoints` is subject to variant handling.] ([RS_SWCT_03148](#))

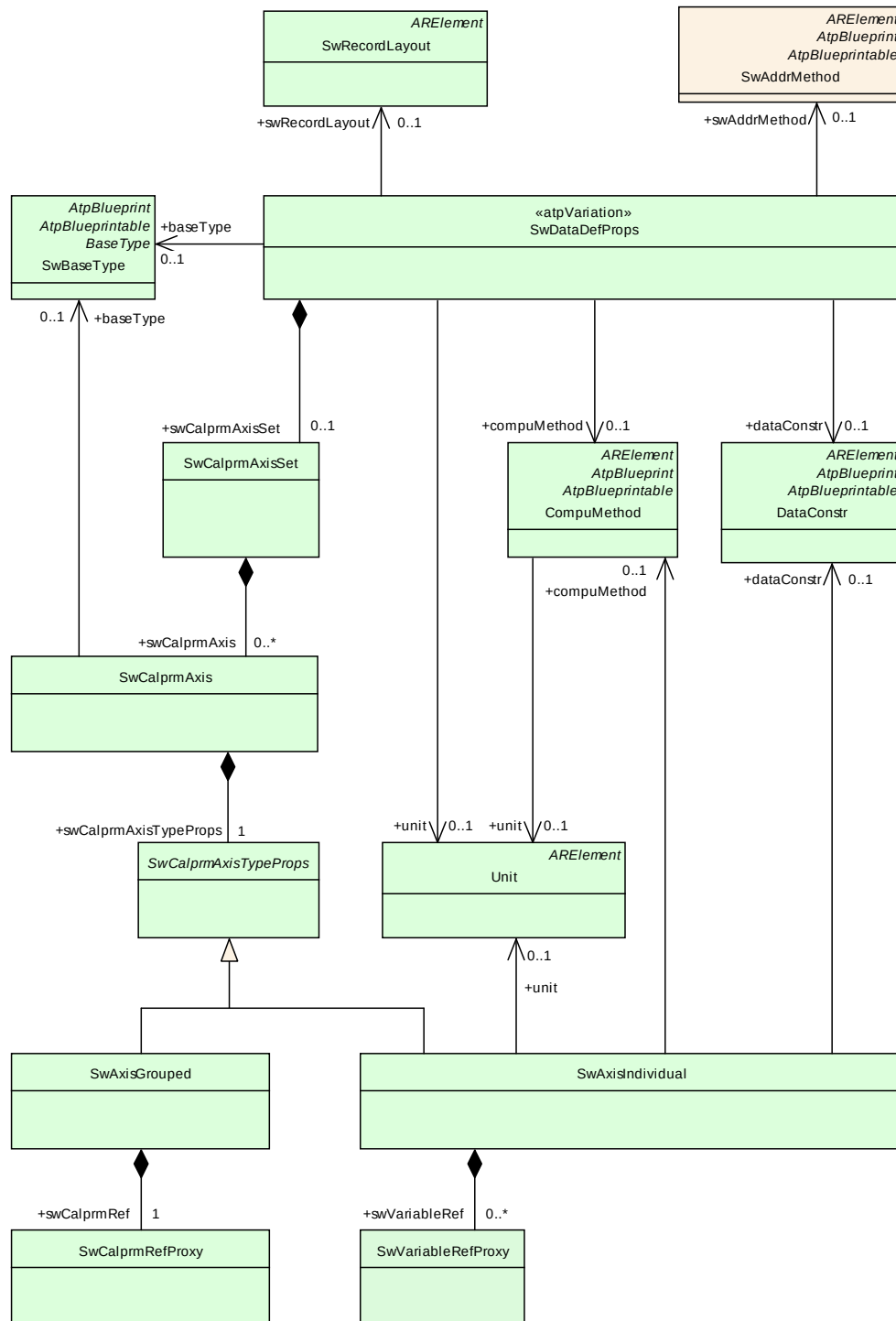


Figure 5.29: Meta-Model Elements used for a Curve

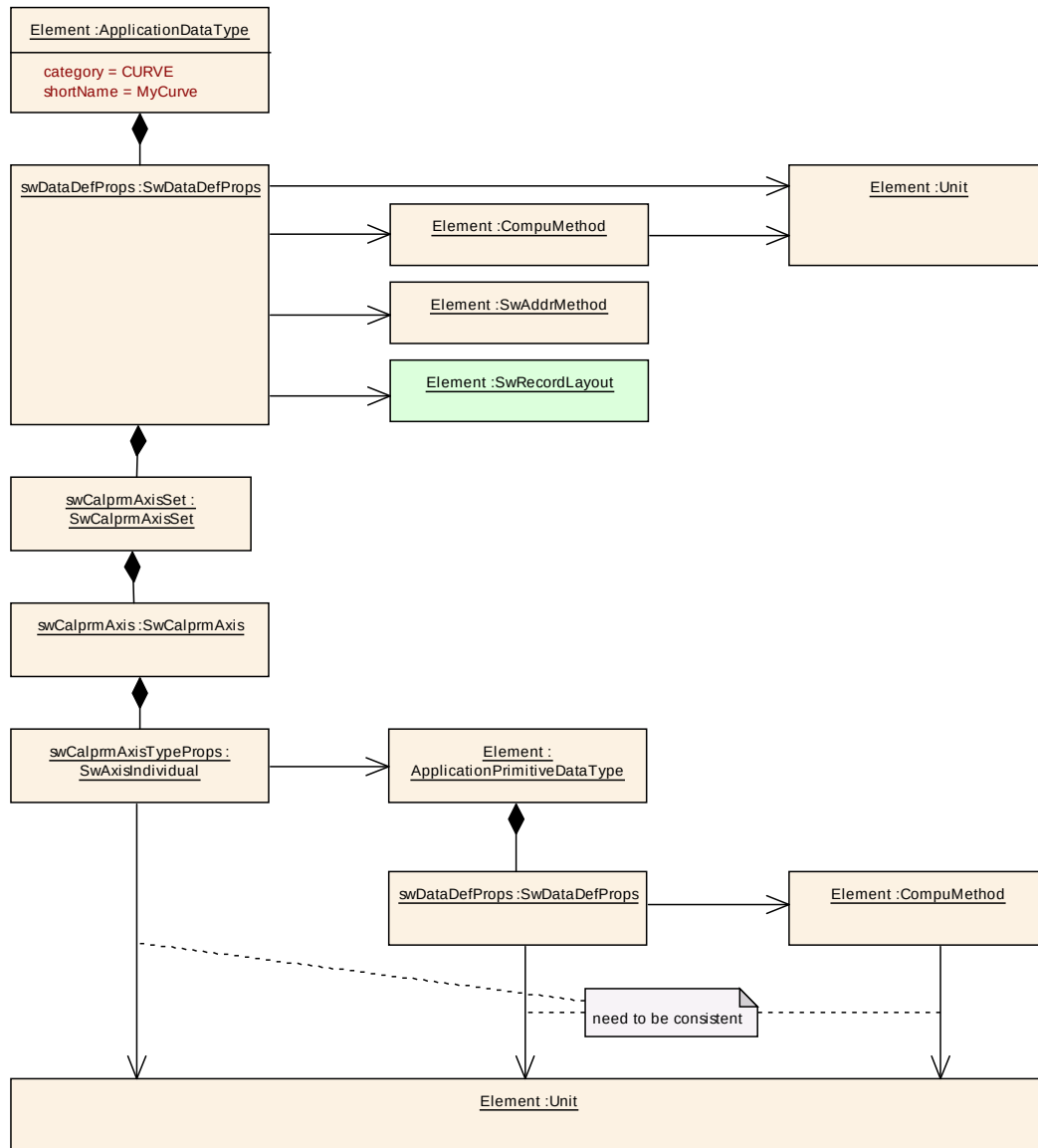


Figure 5.30: Illustration of a Curve in M1

Class	SwCalprmAxisSet			
Package	M2::AUTOSARTemplates::CommonStructure::CalibrationParameter			
Note	This element specifies the input parameter axes (abscissas) of parameters (and variables, if these are used adaptively).			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
swCalprm Axis	SwCalprmAxis	*	aggr	One axis belonging to this SwCalprmAxisSet Tags: xml.roleElement=true; xml.roleWrapperElement=false; xml.sequenceOffset=20; xml.typeElement=false; xml.typeWrapperElement=false

Table 5.50: SwCalprmAxisSet

Class	SwCalprmAxis			
Package	M2::AUTOSARTemplates::CommonStructure::CalibrationParameter			
Note	This element specifies an individual input parameter axis (abscissa).			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
category	CalprmAxisCategoryEnum	0..1	attr	This property specifies the category of a particular axis. Tags: xml.sequenceOffset=30
baseType	SwBaseType	0..1	ref	The SwBaseType to be used for the axis. Note that this is not applicable for ApplicationDataTypes. The value shall be ignored. Tags: atp.Status=removed xml.sequenceOffset=110
displayFormat	DisplayFormatString	0..1	attr	This property specifies how the axis values shall be displayed e.g. in documents or in measurement and calibration tools. Tags: xml.sequenceOffset=100
swAxisIndex	AxisIndexType	0..1	attr	This attribute specifies which axis is specified by the containing SwCalprmAxis. For example in a curve this is usually "1". In a map this is "1" or "2". Tags: xml.sequenceOffset=20
swCalibrationAccess	SwCalibrationAccessEnum	0..1	attr	Describes the applicability of parameters and variables. Tags: xml.sequenceOffset=90
swCalprmAxisTypeProps	SwCalprmAxisTypeProps	1	aggr	specific properties depending on the type of the axis. Tags: xml.roleElement=false; xml.roleWrapperElement=false; xml.sequenceOffset=40; xml.typeElement=true; xml.typeWrapperElement=false

Table 5.51: SwCalprmAxis

Enumeration	CalprmAxisCategoryEnum
Package	M2::AUTOSARTemplates::CommonStructure::CalibrationParameter
Note	This enum specifies the possible values of the category property within SwCalprmAxis.
Literal	Description
comAxis	COM_AXIS is equal to an STD_AXIS, the difference is, that a COM_AXIS is an shared axis, that means this axis can be used multiple times by different CURVES, MAPs, CUBOIDS, CUBE_4s, and CUBE_5s. Tags: xml.name=COM_AXIS

curveAxis	CURVE_AXIS uses a separate CURVE to rescale the axis. The referenced CURVE is used to lookup an axis index, and the index value is used by the controller to determine the operating point in the CURVE, MAP, CUBOID, CUBE_4, or CUBE_5. Tags: xml.name=CURVE_AXIS
fixAxis	FIX_AXIS means that the input axis is not stored. The axis is calculated using parameters and so on it is also not possible to modify the axis points. Tags: xml.name=FIX_AXIS
resAxis	RES_AXIS is also an shared axis like COM_AXIS, the difference is that this kind of axis can be used for rescaling. Tags: xml.name=RES_AXIS
stdAxis	STD_AXIS means that input and output axis definition are stored within this CURVE, MAP, CUBOID, CUBE_4, and CUBE_5. There is no shared or calculated axis. Tags: xml.name=STD_AXIS

Table 5.52: CalprmAxisCategoryEnum

Class	SwCalprmAxisTypeProps (abstract)			
Package	M2::AUTOSARTemplates::CommonStructure::CalibrationParameter			
Note	Base class for the type of the calibration axis. This provides the particular model of the specialization. If the specialization would be the directly from SwCalPrmAxis, the sequence of common properties and the specializes ones would be different.			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
—	—	—	—	—

Table 5.53: SwCalprmAxisTypeProps

Class	SwAxisIndividual			
Package	M2::AUTOSARTemplates::CommonStructure::Axis			
Note	This meta-class describes an axis integrated into a parameter (field etc.). The integration makes this individual to each parameter. The so-called grouped axis represents the counterpart to this. It is conceived as an independent parameter (see class SwAxisGrouped).			
Base	ARObject, SwCalprmAxisTypeProps			
Attribute	Datatype	Mul.	Kind	Note
compuMethod	CompuMethod	0..1	ref	This is the compuMethod which is expected for the axis. It is used in early stages if the particular input-value is not yet available. Tags: xml.sequenceOffset=30
dataConstr	DataConstr	0..1	ref	Refers to constraints, e.g. for plausibility checks. Tags: xml.sequenceOffset=80

Attribute	Datatype	Mul.	Kind	Note
inputVariableType	ApplicationPrimitiveDataType	0..1	ref	This is the datatype of the input value for the axis. This allows to define e.g. a type of curve, where the input value is finalized at the access point.
swAxisGeneric	SwAxisGeneric	0..1	aggr	this specifies the properties of a generic axis if applicable. Tags: xml.sequenceOffset=90
swMaxAxisPoints	Integer	1	attr	Maximum number of base points contained in the axis of a map or curve. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime xml.sequenceOffset=60
swMinAxisPoints	Integer	1	attr	Minimum number of base points contained in the axis of a map or curve. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime xml.sequenceOffset=70
swVariableRef	SwVariableRefProxy	*	aggr	Refers to input variables of the axis. It is possible to specify more than one variable. Here the following is valid: - The variable with the highest priority shall be given first. It is used in the generation of the code and is also displayed first in the application system. - All variables referenced shall be of the same physical nature. This is usually detected in that the conversion formulae affected refer back to the same SI-units. In AUTOSAR this ensured by the constraint, that the referenced input variables shall use a type compatible to "inputVariableType". - This multiple referencing allows a base point distribution for more than one input variable to be used. One example of this are the temperature curves which can depend both on the induction air temperature and the engine temperature. These variables can be displayed simultaneously by MCD systems (adjustment systems), enabling operating points to be shown in the curves. Tags: xml.roleElement=false; xml.roleWrapperElement=true; xml.sequenceOffset=20; xml.typeElement=false; xml.typeWrapperElement=false

Attribute	Datatype	Mul.	Kind	Note
unit	Unit	0..1	ref	This represents the physical unit of the input value of the axis. It is provided to support the case that the particular input variable is not yet known. Tags: xml.sequenceOffset=40

Table 5.54: SwAxisIndividual

Class	SwAxisGeneric			
Package	M2::AUTOSARTemplates::CommonStructure::Axis			
Note	This meta-class defines a generic axis. In a generic axis the axispoints points are calculated in the ECU. The ECU is equipped with a fixed calculation algorithm. Parameters for the algorithm can be stored in the data component of the ECU. Therefore these parameters are specified in the data declaration, not in the calibration data.			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
swAxisType	SwAxisType	1	ref	Associated axis calculation strategy. Tags: xml.sequenceOffset=20
swGenericAxisParam	SwGenericAxisParam	*	aggr	Specific parameter of a generic axis. Tags: xml.roleElement=true; xml.roleWrapperElement=true; xml.sequenceOffset=40; xml.typeElement=false; xml.typeWrapperElement=false

Table 5.55: SwAxisGeneric

Class	SwAxisGrouped			
Package	M2::AUTOSARTemplates::CommonStructure::Axis			
Note	An SwAxisGrouped is an axis which is shared between multiple calibration parameters.			
Base	ARObject, SwCalprmAxisTypeProps			
Attribute	Datatype	Mul.	Kind	Note
sharedAxisType	ApplicationPrimitiveDataType	0..1	ref	This is the datatype of the calibration parameter providing the shared axis.

Attribute	Datatype	Mul.	Kind	Note
swAxisIndex	AxisIndexType	0..1	attr	Describes which axis of the referenced calibration parameter provides the values for the group axis. The index satisfies the following convention: 0 = value axis. in this case, the interpolation result of the referenced parameter is used as a base point index. This means that the A2L keyword CURVE_AXIS_REF can be supported. - The index should only be specified if the parameter under swCalprm contains more than one axis. It is standard practice for the axis index of parameters with more than one axis, to be set to 1, if data has not been assigned to swAxisIndex. Tags: xml.sequenceOffset=20
swCalprmRef	SwCalprmRefProxy	1	aggr	This property specifies the calibration parameter which serves as the input axis. In AUTOSAR, the type of the referenced Calibration parameter shall be compatible to the type specified by sharedAxisType. Tags: xml.roleElement=false; xml.roleWrapperElement=false; xml.sequenceOffset=30; xml.typeElement=false; xml.typeWrapperElement=false

Table 5.56: SwAxisGrouped

5.4.5 Setting an Axis Input Value

When an interpolation routine is called, an input value has to be provided to find the appropriate axis entry in the implementation of a [RunnableEntity](#). However, this input value cannot be arbitrarily chosen but only be selected from available [VariableDataPrototype](#) assigned to it.

In an axis definition attached to an [ApplicationPrimitiveDataType](#), it is possible to specify the [inputVariableType](#) for the input values.

[constr_1019] Compatibility of input value and axis [The [SwDataDefProps](#) the input variable shall be compatible to the [datatype](#) resp. [compuMethod](#) resp. [unit](#) of the [SwAxisIndividual](#).]()

Every [ParameterDataPrototype](#) then allows to specify zero or more input values (being type compatible to [inputVariableType](#)) in its axis description.

This means that at the specification time of an [SwcInternalBehavior](#) a list of input values has to be specified where the implementer of a [RunnableEntity](#) can choose of. The input values are [DataPrototype](#) entities either being

- a `VariableDataPrototype` in a `SenderReceiverInterface` or `NvDataInterface` of a `PortPrototype`, of the `AtomicSwComponentType` where the `SwcInternalBehavior` is associated to, or an `ArgumentDataPrototype` in a `ClientServerOperation` of a `ClientServerInterface` in a `PortPrototype` of the `AtomicSwComponentType` where the `InternalBehavior` is associated to, or
- an `VariableDataPrototype` within the `SwcInternalBehavior`.

To achieve this, `SwAxisIndividual` is aggregating a `SwVariableRefProxy`.

Originally, MSRSW uses a `AutosarVariableRef` to set the input value of an axis appropriately. In AUTOSAR, this has been extended by first introducing a `SwVariableRefProxy`.

Note that this is a specific use case for the role `SwVariableRefProxy.autosarVariable`.

Note further that the use cases for the existence of the attributes `SwVariableRefProxy.autosarVariable` and `SwVariableRefProxy.mcDataInstanceVar` are entirely disjoint and therefore the simultaneous existence of these two attributes would not make any sense at all.

Therefore, `[constr_1382]` has been introduced to clarify this aspect.

`[constr_1382]` Mutually exclusive existence of attributes `SwVariableRefProxy.autosarVariable` vs. `SwVariableRefProxy.mcDataInstanceVar` [In any given AUTOSAR model, the aggregations `SwVariableRefProxy.autosarVariable` and `SwVariableRefProxy.mcDataInstanceVar` shall never exist at the same time.]()

As shown in Figure 5.31, this approach is also used to represent a `AutosarVariableRef` in all roles, e.g. the result of an interpolation routine applied to an axis, the input value determination, a list of dependent parameters, and `swDataDependency`.

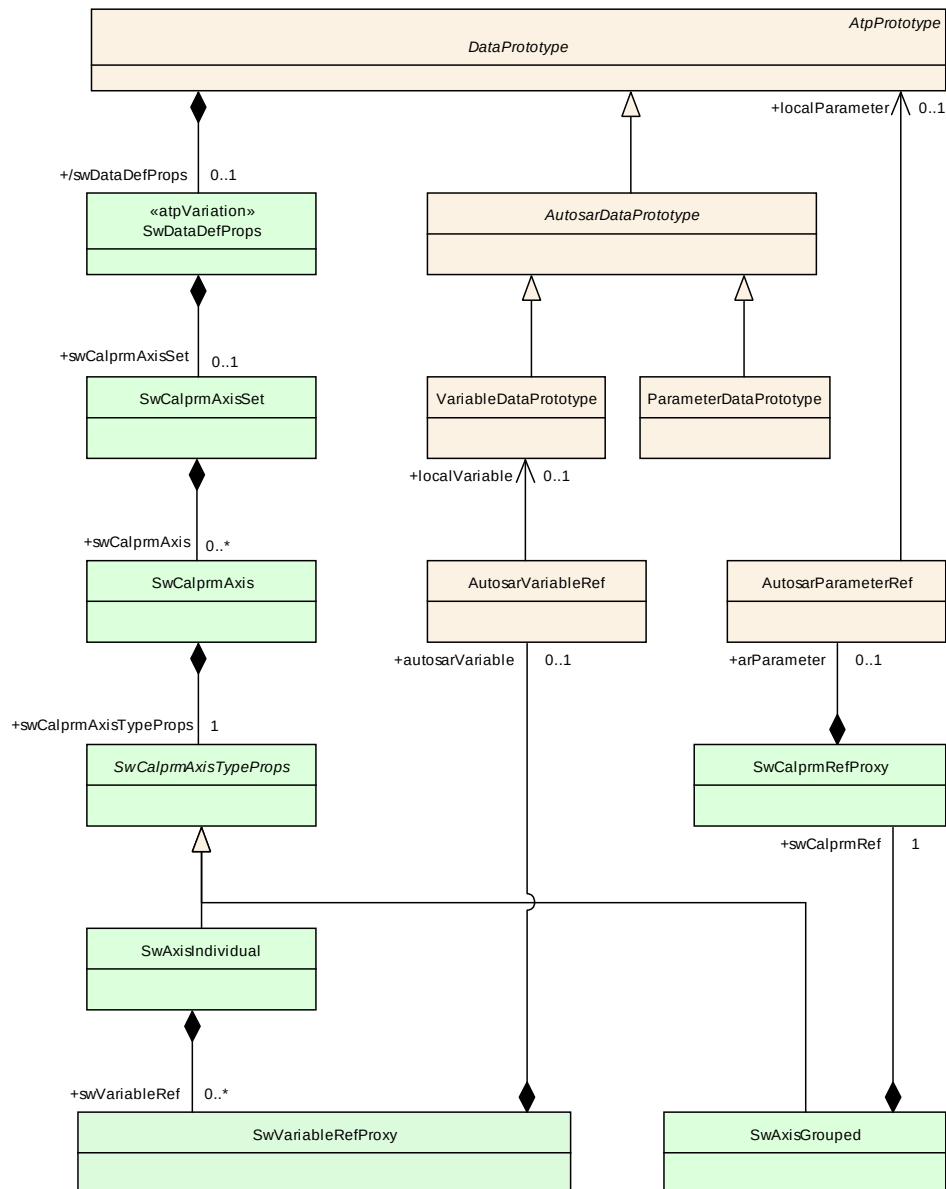


Figure 5.31: Extended Axis Elements and Input Variable Reference

Grouped curves share the same axis definition. In MSRSW, this is shown by referencing the `SwCalprm`, representing an individual curve, from a `SwAxisGrouped`.

Note that this does not describe which axis shall be taken from a reference `swCalprmRef` acting as a shared axis. This would be done in `SwAxisGrouped.swAxisIndex`.

AUTOSAR applies a similar proxy approach for parameters as for the variables. Therefore, an `SwCalprmRefProxy` has been introduced in MSRSW, and is aggregated by the `SwAxisGrouped` element.

The `SwCalprmRefProxy` aggregates an `AutosarParameterRef` providing an association to a `ParameterDataPrototype`, representing a curve with an axis. When defining the data type of a parameter the type of the shared axis is defined in `sharedAxisType`.

[constr_1020] **ParameterDataPrototype** needs to be of compatible data type as referenced in **sharedAxisType** [Finally, the **ParameterDataPrototype** assigned in **swCalprmRef** shall be typed by data type compatible to **sharedAxisType**.]()

The AUTOSAR-style is shown in the upper left part of Figure 5.31, while in the upper middle the MSRSW style is shown, referencing the **SwCalprm**.

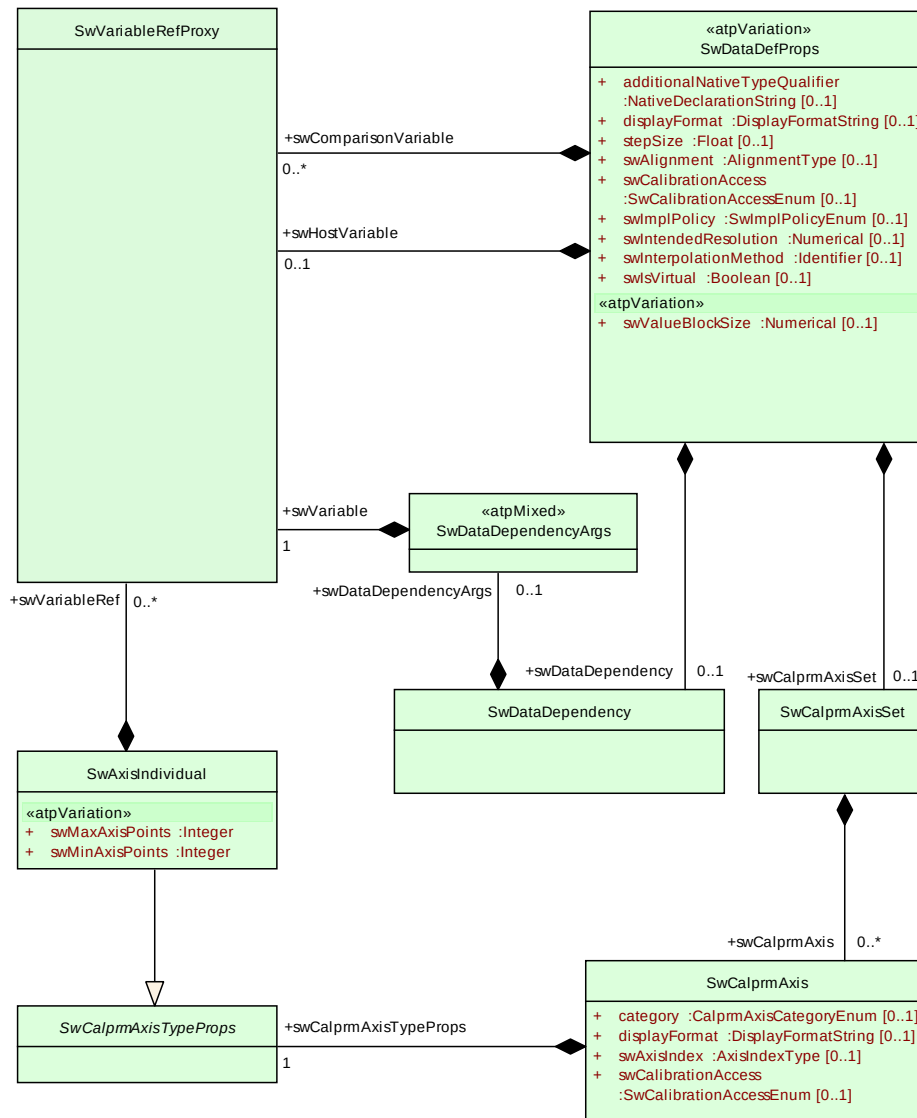


Figure 5.32: Applying Proxy Variable Reference Mechanism

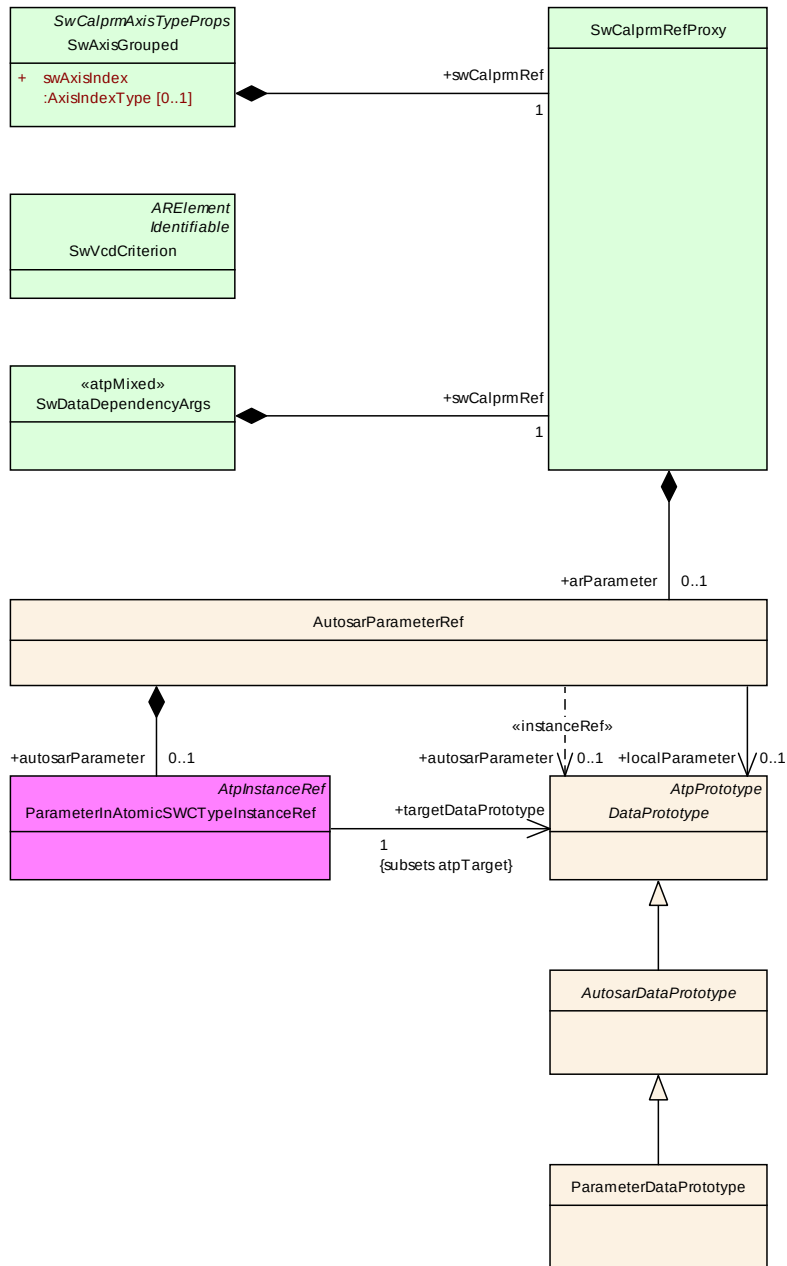


Figure 5.33: Applying Proxy Parameter Reference Mechanism

Class	SwCalprmRefProxy			
Package	M2::AUTOSARTemplates::CommonStructure::DatadictionaryProxies			
Note	Wrapper class for different kinds of references to a calibration parameter.			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
arParameter	AutosarParameterRef	0..1	aggr	This represents a Parameter within AUTOSAR. Note that the Datatype of the referenced ParameterDataPrototype shall be an ApplicationDataType of category VALUE.

Attribute	Datatype	Mul.	Kind	Note
mcDataInstance	McDataInstance	0..1	ref	This reference is used in the McSupport file to express the final instance of group axis etc. It is not allowed to use this outside of an McDataInstance. The referenced mcDataInstance shall be originated from a ParameterDataPrototype.

Table 5.57: SwCalprmRefProxy

Class	SwVariableRefProxy			
Package	M2::AUTOSARTemplates::CommonStructure::DatadictionaryProxies			
Note	Proxy class for several kinds of references to a variable.			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
autosarVariable	AutosarVariableRef	0..1	aggr	This represents the reference to a Variable in an Autosar system. Note that the target of the reference within AutosarVariableRef shall be typed by a primitive data type
mcDataInstanceVar	McDataInstance	0..1	ref	This reference is used in the McSupport file to express the final instance of input values etc. It is not allowed to use this outside of an McDataInstance. The referenced mcDataInstance shall be originated from a VariableDataPrototype.

Table 5.58: SwVariableRefProxy

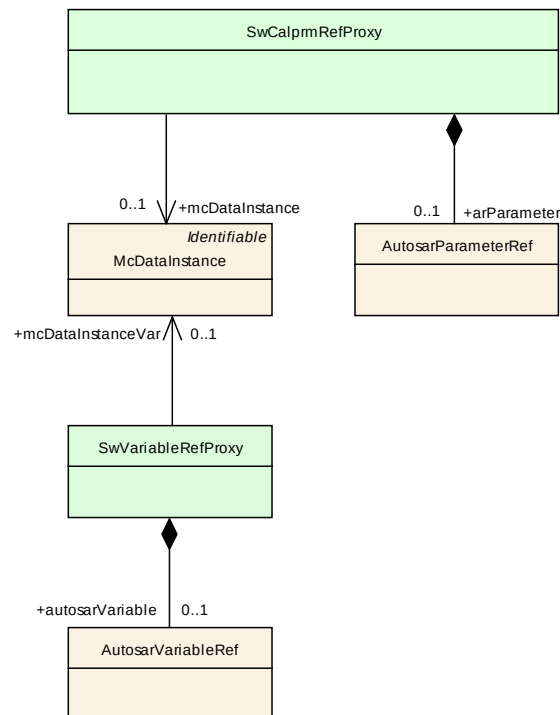


Figure 5.34: Proxy reference classes

The basic patterns for referencing [DataPrototypes](#) are explained in section 5.3.2. In the context of this chapter it is worth to remark that the definition of access to calibration parameters is implemented in the context of a [RunnableEntity](#) (see Figure 7.3).

As the definition of a calibration parameter may involve the definition of several axes the necessity to provide this amount of information might become cumbersome and (to some extent) redundant and difficult to maintain if the same calibration parameter is accessed from within several [RunnableEntity](#)s. In other words: in this case it would be necessary to repeat the more or less complex set of information for each [RunnableEntity](#).

To avoid this unnecessary level of complexity for the definition of access to calibration parameters, it is possible to define the access to the calibration parameter on the level of [InstantiationDataDefProps](#) which have been defined to facilitate this kind of re-use (for more information please refer to section 7.5.4). This ability is also documented in Table 5.39.

5.4.6 Specifying Data Dependencies

[SwDataDependency](#) allows dependent data elements to be specified. For example, other [ParameterDataPrototypes](#) can be combined into one [ParameterDataPrototype](#) whose consistent value is automatically derived by the measurement and calibration system. Upon adjusting one of the parameters, the dependent parameter is then also automatically adjusted according to the chosen formula.

Consider for example a rectangular triangle with a hypotenuse of length 1, where the length of the other sides are the parameter A and B. When adjusting A the parameter B has to be adjusted accordingly to $B = \sqrt{(1 - A * A)}$. Also other parameters might depend on B, e.g. $B\_AREA = B * B$ or $TRIANGULAR\_AREA = (A * B)/2$. This example is shown in listing 5.6.

A dependent parameter should not be adjustable by itself. The only way to influence its value is through the adjustment of a parameter it depends on.

Listing 5.6: Data Dependency

```
<PER-INSTANCE-PARAMETERS>
  <PARAMETER-DATA-PROTOTYPE>
    <SHORT-NAME>A</SHORT-NAME>
    <DESC>
      <L-2 L="DE">The independent Parameter</L-2>
    </DESC>
    <CATEGORY>VALUE</CATEGORY>
  </PARAMETER-DATA-PROTOTYPE>
  <PARAMETER-DATA-PROTOTYPE>
    <SHORT-NAME>B</SHORT-NAME>
    <DESC>
      <L-2 L="DE">The dependent Parameter</L-2>
    </DESC>
    <SW-DATA-DEF-PROPS>
      <SW-DATA-DEF-PROPS-VARIANTS>
        <SW-DATA-DEF-PROPS-CONDITIONAL>
          <SW-DATA-DEPENDENCY>
            <SW-DATA-DEPENDENCY-FORMULA>SQRT ( X1 * X1 )</SW-
              DATA-DEPENDENCY-FORMULA>
            <SW-DATA-DEPENDENCY-ARGS>
              <AR-PARAMETER>
                <LOCAL-PARAMETER-REF DEST="PARAMETER-DATA-
                  PROTOTYPE">/DataDependency/foo/bar/A</
                  LOCAL-PARAMETER-REF>
              </AR-PARAMETER>
            </SW-DATA-DEPENDENCY-ARGS>
          </SW-DATA-DEPENDENCY>
        </SW-DATA-DEF-PROPS-CONDITIONAL>
      </SW-DATA-DEF-PROPS-VARIANTS>
    </SW-DATA-DEF-PROPS>
  </PARAMETER-DATA-PROTOTYPE>
  <PARAMETER-DATA-PROTOTYPE>
    <SHORT-NAME>B_AREA</SHORT-NAME>
    <DESC>
      <L-2 L="DE">The dependent Parameter</L-2>
    </DESC>
    <SW-DATA-DEF-PROPS>
      <SW-DATA-DEF-PROPS-VARIANTS>
        <SW-DATA-DEF-PROPS-CONDITIONAL>
          <SW-DATA-DEPENDENCY>
            <SW-DATA-DEPENDENCY-FORMULA>X1 * X1</SW-DATA-
              DEPENDENCY-FORMULA>
            <SW-DATA-DEPENDENCY-ARGS>
              <AR-PARAMETER>
```

```

        <LOCAL-PARAMETER-REF DEST="PARAMETER-DATA-
            PROTOTYPE">/DataDependency/foo/bar/B</
            LOCAL-PARAMETER-REF>
        </AR-PARAMETER>
    </SW-DATA-DEPENDENCY-ARGS>
</SW-DATA-DEPENDENCY>
</SW-DATA-DEF-PROPS-CONDITIONAL>
</SW-DATA-DEF-PROPS-VARIANTS>
</SW-DATA-DEF-PROPS>
</PARAMETER-DATA-PROTOTYPE>
<PARAMETER-DATA-PROTOTYPE>
    <SHORT-NAME>TRIANGULAR_AREA</SHORT-NAME>
    <DESC>
        <L-2 L="DE">The dependent Parameter</L-2>
    </DESC>
    <SW-DATA-DEF-PROPS>
        <SW-DATA-DEF-PROPS-VARIANTS>
            <SW-DATA-DEF-PROPS-CONDITIONAL>
                <SW-DATA-DEPENDENCY>
                    <SW-DATA-DEPENDENCY-FORMULA>(X1 * X2) / 2</SW-
                        DATA-DEPENDENCY-FORMULA>
                    <SW-DATA-DEPENDENCY-ARGS>
                        <AR-PARAMETER>
                            <LOCAL-PARAMETER-REF DEST="PARAMETER-DATA-
                                PROTOTYPE">/DataDependency/foo/bar/A</
                                    LOCAL-PARAMETER-REF>
                        </AR-PARAMETER>
                        <AR-PARAMETER>
                            <LOCAL-PARAMETER-REF DEST="PARAMETER-DATA-
                                PROTOTYPE">/DataDependency/foo/bar/B</
                                    LOCAL-PARAMETER-REF>
                        </AR-PARAMETER>
                    </SW-DATA-DEPENDENCY-ARGS>
                </SW-DATA-DEPENDENCY>
            </SW-DATA-DEF-PROPS-CONDITIONAL>
        </SW-DATA-DEF-PROPS-VARIANTS>
    </SW-DATA-DEF-PROPS>
</PARAMETER-DATA-PROTOTYPE>
</PER-INSTANCE-PARAMETERS>
</SWC-INTERNAL-BEHAVIOR>
</INTERNAL-BEHAVIORS>

```

Class	SwDataDependency			
Package	M2::AUTOSARTemplates::CommonStructure::DataDefProperties			
Note	This element describes the interdependencies of data objects, e.g. variables and parameters. Use cases: - Calculate the value of a calibration parameter (by the MCD system) from the value(s) of other calibration parameters. - Virtual data - that means the data object is not directly in the ecu and this property describes how the "virtual variable" can be computed from the real ones (by the MCD system).			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
swDataDependencyArgs	SwDataDependencyArgs	0..1	aggr	Specifies the arguments used in the data dependency. Note that this is 0..1 since the aggregated class is a container (atpMixed). Tags: xml.sequenceOffset=40
swDataDependencyFormula	CompuGenericMath	0..1	aggr	This element describes the formula with which the dependencies between the participating objects are defined. Tags: xml.sequenceOffset=30

Table 5.59: SwDataDependency

Class	«atpMixed» SwDataDependencyArgs			
Package	M2::AUTOSARTemplates::CommonStructure::DataDefProperties			
Note	This element specifies the elements used in a SwDataDependency.			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
swCalprmRef	SwCalprmRefProxy	1	aggr	Specifies a calibration parameter as an input argument to the dependency. Tags: xml.roleElement=false; xml.roleWrapperElement=false; xml.sequenceOffset=60; xml.typeElement=false; xml.typeWrapperElement=false
swVariable	SwVariableRefProxy	1	aggr	Specifies a variable as an input argument to the dependency. Tags: xml.roleElement=false; xml.roleWrapperElement=false; xml.sequenceOffset=70; xml.typeElement=false; xml.typeWrapperElement=false

Table 5.60: SwDataDependencyArgs

5.4.7 Precedence of data properties with respect to data elements, axis elements, computation methods, units

There are similar attributes defined in `SwDataDefProps` as well as in `SwCalprmAxis` as well as in `CompuMethod`. Therefore we need to define which attribute value wins in the overall process from SWC-Description to MC-Support to ASAM-A2L.

Figure 5.35 illustrates the fact that some attributes in `SwDataDefProps` can also be expressed in subelements respectively in referenced elements.

[TPS_SWCT_01496] General precedence rule for attributes of `SwDataDefProps`

[The general precedence rule is that

- `SwDataDefProps` wins over `valueAxisDataType` (exception: `compuMethod` and `unit`).
- `SwDataDefProps` wins over `compuMethod`.
- `SwDataDefProps` wins over `swCalprmAxisSet`.
- `SwDataDefProps.swCalprmAxisSet` wins over `swCalprmAxisSet.swCalprmAxis.swCalprmAxisTypeProps.compuMethod` resp. `SwAxisIndividual.inputVariableType`.
- `SwAxisIndividual.inputVariableType` wins over `SwAxisIndividual.compuMethod`, `SwAxisIndividual.unit`, but **not** over `SwAxisIndividual.dataConstr`.

]()

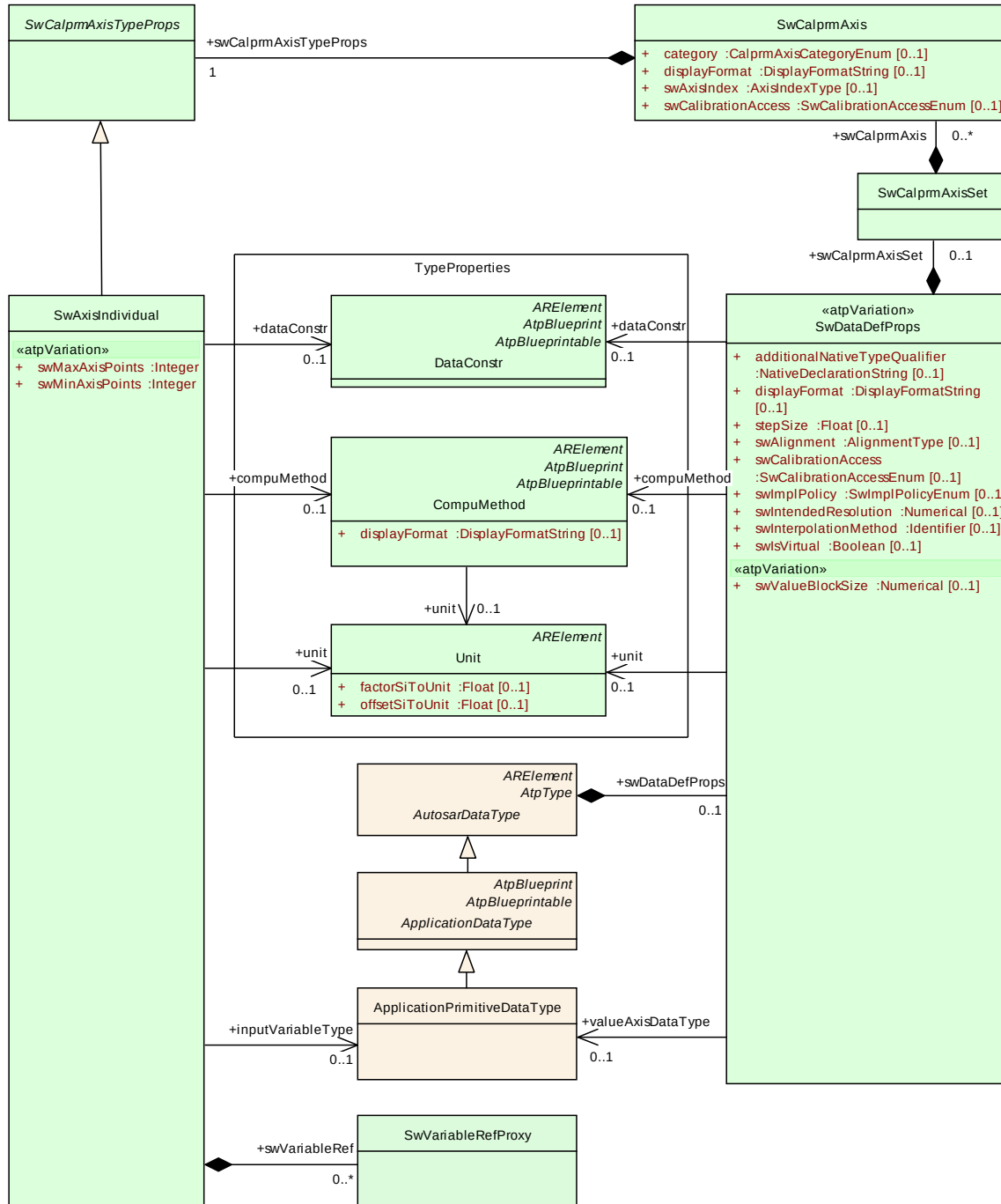


Figure 5.35: Various Attributes in the Context of **SwDataDefProps**

The following examples illustrate particular cases (the highest precedence comes first):

- **[TPS_SWCT_01497] Precedence of the unit of value axis** [For the usage of unit of value axis the following precedence rule is defined:
 - `SwDataDefProps.valueAxisDataType.swDataDefProps.unit`
 - `SwDataDefProps.valueAxisDataType.swDataDefProps.compuMethod.unit`

- `SwDataDefProps.unit`
- `SwDataDefProps.compuMethod.unit`

⌋()

[constr_2550] Units of value axis shall be consistent ⌈ The units specified in the context of value axis shall be the same, even if there is a precedence rule. ⌋()

In particular, [\[constr_2550\]](#) reflects the fact that `unit` may be specified in different phases of the development process but finally need to be consistent.

- **[TPS_SWCT_01498] Precedence of the `DataConstr` of value axis** ⌈ For the usage of `DataConstr` of value axis the following precedence rule is defined:

- `SwDataDefProps.dataConstr`
- `SwDataDefProps.valueAxisDataType.swDataDefProps.dataConstr`

⌋()

[constr_2548] Data constraint of value axis shall match ⌈ The values compliant to `SwDataDefProps.dataConstr` shall be also be compliant to `SwDataDefProps.valueAxisDataType.swDataDefProps.dataConstr`.

In other words `SwDataDefProps.dataConstr` win over but are not allowed to relax `SwDataDefProps.valueAxisDataType.swDataDefProps.dataConstr` but are not allowed ⌋()

- **[TPS_SWCT_01499] Precedence of the `CompuMethod` of value axis** ⌈ For the usage of `CompuMethod` of value axis the following precedence rule is defined:

- `SwDataDefProps.valueAxisDataType.swDataDefProps.compuMethod`
- `SwDataDefProps.compuMethod`

⌋()

- **[TPS_SWCT_01500] Precedence of the display format of value axis** ⌈ For the usage of display format of value axis the following precedence rule is defined:

- `SwDataDefProps.displayFormat`
- `SwDataDefProps.valueAxisDataType.swDataDefProps.displayFormat`
- `SwDataDefProps.valueAxisDataType.swDataDefProps.compuMethod.displayFormat`
- `SwDataDefProps.compuMethod.displayFormat`

⌋()

Note that this deviates from the general rule since `displayFormat` is not an essential property. The last item in the list above is the consequence of the fact that if there is a `valueAxisDataType` it supersedes the `compuMethod`

- **[TPS_SWCT_01501] Precedence of the calibration access of value axis** [For the usage of calibration access of value axis the following precedence rule is defined:

- `SwDataDefProps.swCalibrationAccess`
- `SwDataDefProps.valueAxisDataType.swDataDefProps.swCalibrationAccess`

]()

Note that this deviates from the general rule since `swCalibrationAccess` is not such an essential property.

- **[TPS_SWCT_01502] Precedence of the Unit of the input axis** [For the usage of `Unit` of the input axis the following precedence rule is defined:

- `SwAxisIndividual.unit`
- `SwAxisIndividual.compuMethod.unit`
- `SwAxisIndividual.inputVariableType.swDataDefProps.unit`
- `SwAxisIndividual.swVariableRef.autosarVariable.autosarVariable.type.swDataDefProps.compuMethod.unit`
- `SwAxisIndividual.swVariableRef.autosarVariable.autosarVariable.type.swDataDefProps.unit`

]()

[constr_2549] Units of input axis shall be consistent [The units specified in the context of an input axis shall be compatible, even if there is a precedence rule.]()

[[constr_2549](#)] reflects the fact that `unit` may be specified in different phases of the development process but finally need to be consistent.

- **[TPS_SWCT_01503] Precedence of the DataConstr of the input axis** [For the usage of `DataConstr` of the input axis the following precedence rule is defined:

- `SwAxisIndividual.dataConstr`
- `SwAxisIndividual.inputVariableType.swDataDefProps.dataConstr`
- `SwAxisIndividual.swVariableRef.type.swDataDefProps.dataConstr`

]()

Note that `SwAxisIndividual.inputVariableType.swDataDefProps.dataConstr` represent the input value, not the axis itself. For this reason there is no specific constraint that the `dataConstr` need to match.

- **[TPS_SWCT_01504] Precedence of the display format of the input axis** [For the usage of display format of the input axis the following precedence rule is defined:

- `SwCalprmAxis.displayFormat`
- `SwCalprmAxis.swCalprmAxisTypeProps.compuMethod.displayFormat`
- `SwCalprmAxis.swCalprmAxisTypeProps.inputVariableType.swDataDefProps.displayFormat`
- `SwCalprmAxis.swCalprmAxisTypeProps.inputVariableType.swDataDefProps.compuMethod.displayFormat`
- `SwCalprmAxis.swCalprmAxisTypeProps.swVariableRef.type.swDataDefProps.displayFormat`
- `SwCalprmAxis.swCalprmAxisTypeProps.swVariableRef.type.swDataDefProps.compuMethod.displayFormat`

]()

Please note that `SwAxisIndividual.inputVariableType.swDataDefProps.dataConstr` represent the input value and not the axis itself. For this reason there is no specific constraint that `displayFormat` needs to match.

- **[TPS_SWCT_01505] Precedence of calibration access along structure hierarchies in complex types** [For the usage of calibration access along structure hierarchies in complex types the precedence rule is defined in table 5.61.]()

outer	inner	result
<code>notAccessible</code>	<code>*</code>	<code>notAccessible</code>
<code>readOnly</code>	<code>readOnly</code>	<code>readOnly</code>
<code>readOnly</code>	<code>readWrite</code>	<code>readOnly</code>
<code>readOnly</code>	<code>notAccessible</code>	<code>notAccessible</code>
<code>readWrite</code>	<code>notAccessible</code>	<code>notAccessible</code>
<code>readWrite</code>	<code>readOnly</code>	<code>readOnly</code>
<code>readWrite</code>	<code>readWrite</code>	<code>readWrite</code>

Table 5.61: Precedence of `swCalibrationAccess` along structure hierarchies

The interpretation of table 5.61 is it lists possible combinations of values of `SwCalibrationAccessEnum` for outer and inner elements of a complex data type and the (in the column "result") indicates value of `SwCalibrationAccessEnum` applicable for this specific combination.

- **[TPS_SWCT_01506] Precedence of the calibration access of input axis** [For the usage of calibration access of input axis the following precedence rule is defined:

- `SwDataDefProps.swCalibrationAccess`
- `SwCalprmAxis.swCalibrationAccess`

]()

Note that the `swCalibrationAccess` defined on a Compound Primitive Data Type (see [TPS_SWCT_01179]) reflects the entire curve or map.

Therefore, if the entire curve or map cannot be accessed by the measurement calibration diagnostic system (MCD-System), the axis can also not be accessed. On the other hand it might be that access is granted for the value axis only but not for the axis points.

5.5 Elements used in Properties of Data Definitions

This section describes further elements which are attached to `SwDataDefProps` via associations.

5.5.1 Computation Methods

[TPS_SWCT_01276] Computation methods [An important part of semantics is the specification of a so-called computation method which specifies the conversion between the physical and the internal representation of data. This usually makes sense only for primitive data types.]()

An `ApplicationCompositeDataType` cannot be given a particular semantic meaning as a whole but it is obviously possible to specify the semantics of all or a part of the contained elements, i.e. the `ApplicationPrimitiveDataTypes`.

Class	CompuMethod			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::ComputationMethod			
Note	This meta-class represents the ability to express the relationship between a physical value and the mathematical representation. Note that this is still independent of the technical implementation in data types. It only specifies the formula how the internal value corresponds to its physical pendant. Tags: atp.recommendedPackage=CompuMethods			
Base	<code>ARElement</code> , <code>ARObject</code> , <code>AtpBlueprint</code> , <code>AtpBlueprintable</code> , <code>CollectableElement</code> , <code>Identifiable</code> , <code>MultilanguageReferrable</code> , <code>PackageableElement</code> , <code>Referrable</code>			
Attribute	<i>Datatype</i>	<i>Mul.</i>	<i>Kind</i>	<i>Note</i>

Attribute	Datatype	Mul.	Kind	Note
compuInternalToPhys	Compu	0..1	aggr	This specifies the computation from internal values to physical values. Tags: xml.sequenceOffset=80
compuPhysToInternal	Compu	0..1	aggr	This represents the computation from physical values to the internal values. Tags: xml.sequenceOffset=90
displayFormat	DisplayFormatString	0..1	attr	This property specifies, how the physical value shall be displayed e.g. in documents or measurement and calibration tools. Tags: xml.sequenceOffset=20
unit	Unit	0..1	ref	This is the physical unit of the Physical values for which the CompuMethod applies. Tags: xml.sequenceOffset=30

Table 5.62: CompuMethod

This meta-class `CompuMethod` was actually taken from the *ASAM* standard's *harmonized data objects*. This is also indicated by the green color of the meta-classes in the diagram.

[constr_1142] category of `CompuMethod` shall not be extended [In contrast to the general rule that `category` can be extended by user-specific values it is **not allowed** to extend the meaning of the attribute `category` of meta-class `CompuMethod`]()

[TPS_SWCT_01277] Computation methods are used for the conversion of *internal* values into their *physical* representation and vice versa [`CompuMethods` (see Figure 5.36) are used for the conversion of *internal* values into their *physical* representation and vice versa. The direction of the conversion depends on the origin of the value to be converted:

- If the value is provided by the ECU then the conversion direction is from internal to physical.
- If a physical value is provided by the tester it is converted to internal values before being sent to the ECU

]()

[TPS_SWCT_01548] Limits of a `CompuMethod` [In case `CompuScale.lowerLimit` and `CompuScale.upperLimit` are used to constrain the applicable range of the conversion of a `CompuMethod`, they logically represent the limiting values **before** the conversion is applied.]()

In other words, the limits are applied on the source end of the conversion rather than to the result that comes out at the other end of the conversion. This is obviously a lot safer than the opposite approach where a given physical/internal value would first be converted to its internal/physical equivalent and then, after the conversion is finished

there would be (as a second step) the obligation to check whether the result of the conversion is actually valid in terms of the applicable limits.

[TPS_SWCT_01278] CompuMethods can also be used to assign symbolic names to internal values [CompuMethods can also be used to assign symbolic names to internal values (like an enumeration in C) or to ranges of internal values or to single bits (like a bitfield in C). This is also considered as a conversion between internal numbers and a semantical representation. Some examples are given below.]()

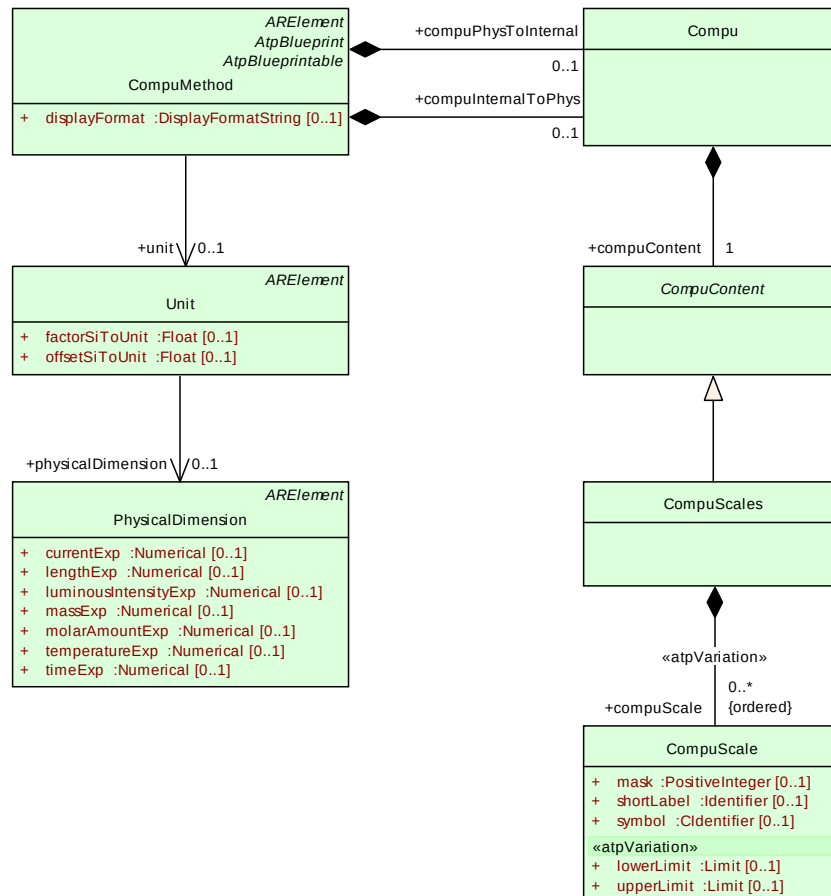


Figure 5.36: A **CompuMethod** and its attributes define data semantics

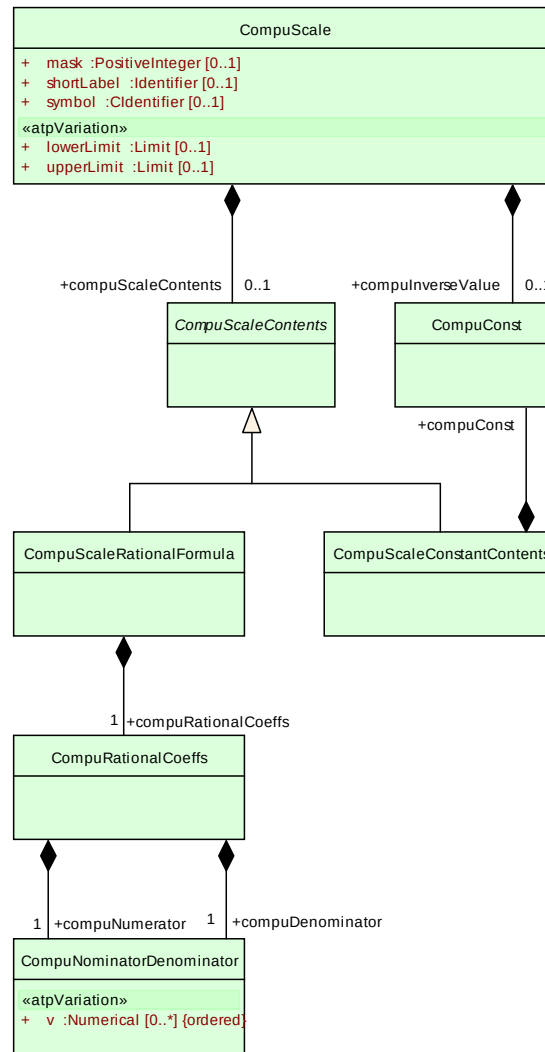


Figure 5.37: A **CompuScale and its attributes define data semantics**

[TPS_SWCT_01279] Preferred conversion direction depends on the use case [

The preferred conversion direction depends on the use case. The physical-to-internal direction is suitable for calibration while the internal-to-physical direction is preferred for diagnostic purposes.]()

In the following, the internal-to-physical conversion direction is used as the default. Usually a **CompuMethod** is defined for one conversion direction only even if it is used in both directions.

For simple functions like identical (1:1 conversion) or linear functions this is sufficient because the inverse function can be derived quite easily from the defined function. In this case also the limits for the reverse direction can be gained by applying the forward function to the forward limits.

For more complex functions (e.g. rational functions) it is usually not possible to compute the inverse function automatically. More seriously, the inversion yields ambiguous results if the function is not monotonic. To deal with such possible ambiguities in a

direct way an inverse value can be provided explicitly for the function or for each of its parts respectively.

[constr_1021] A `CompuMethod` shall specify instructions for both directions [The forward and inverse direction shall always be clearly determined either by

- explicitly specifying both directions
- automatically inverting the `CompuMethod` if applicable

]()

[constr_1022] Limits shall be defined for each direction of `CompuMethod` [In case that both domains are specified in the `CompuMethod` both shall have explicitly defined limits.]()

[TPS_SWCT_01280] `CompuMethod` applied to values outside of its limits [If a `CompuMethod` is applied to values outside of its limits, it is up to the MCD-tool (Measurement, Calibration, Diagnostic tool) to indicate this to the user. In this case the `CompuMethod` shall not be applied at all.]()

[constr_1175] Depending on its `category`, `CompuMethod` shall refer to a `unit` [As a `CompuMethod` specifies the conversion between the physical world and the numerical values they shall refer to a `unit` unless the `CompuMethod`'s `category` is one of `TEXTTABLE`, `BITFIELD_TEXTTABLE`, or `IDENTICAL`.]()

[**constr_1175**] does *not* imply that `CompuMethods` where the `category` is one of `TEXTTABLE`, `BITFIELD_TEXTTABLE`, or `IDENTICAL` are not *allowed* to refer to a unit. They may still refer to a `unit`, but according to [**constr_1175**] this relation is not *mandated*.

A further implication is that the unit itself may not have a dimension, i.e. all exponents of SI units are 0.

Figure 5.36 sketches a conceptual overview of `CompuMethod`. It consists of the following attributes:

- **[TPS_SWCT_01281] `Unit` associated with a `PhysicalDimension`** [A unit (described in next section) can be associated with a `PhysicalDimension`.]()

Note that quantities like “%” are not derived from SI units. However, they have a meaning in the physical world and need to be represented in form of data types. Therefore, a `CompuMethod` also applies in those cases.

- **[TPS_SWCT_01430] Conversion specification from internal to physical values as well as the reverse conversion** [A conversion specification from internal to physical values, as well as the reverse conversion. Both of them in turn consist of an abstract `CompuContent`. Derived classes allow the specification of a conversion formula in two different ways.]()

[constr_1024] Stepwise definition of `CompuMethods` [Within AUTOSAR only the stepwise definition (`CompuScales`) is used.]()

- **[TPS_SWCT_01282] Number of intervals in which a given conversion applies** [CompuScales is a number of intervals (called CompuScale) within which a certain conversion applies. The respective interval is given in terms of upper and lower limit. Limits are explained in more detail in chapter 5.2.4.1.]()

Within each CompuScale we have the abstract CompuScaleContents. To deal with possible ambiguities in a direct way an inverse value can be provided explicitly for that particular scale (compuInverseValue).]()

- As the diagram shows, CompuScaleContents is an abstract meta-class. A number of derived meta-classes allow the specification of a conversion formula in a variety of ways, including:
 - mapping the whole interval to a constant (CompuConst)
 - providing rational coefficients of the conversion formula (CompuRational-Coeffs)

- **[TPS_SWCT_01283] Rational function** [The rational function is specified as rational coefficients for the numerator (compuNumerator) and the denominator (compuDenominator). CompuNominatorDenominator can have as many V elements as needed for the rational function.]()

The sequence of the values V carries the information for the exponents, that means the first V is the coefficient for x0, the second V is the coefficient for x1, etc. With this sequence the values of the exponents can be entirely represented.]()

[constr_1025] Avoid division by zero in rational formula [The rational formula shall not yield any division by zero.]()

[TPS_SWCT_01284] CompuScale might require a representation in the generated RTE C code [A CompuScale might require a representation in the generated RTE C code. For this purpose it is necessary to identify a property that controls how to symbol used for the CompuScale in the C code is created. The symbol itself can be created out of different sources according to a standardized precedence schema.]()

[TPS_SWCT_01569] Definition of CompuScale Symbolic Name [In C code, a CompuScale is represented by an identifier that is, as far as AUTOSAR modeling is concerned, called a CompuScale Symbolic Name. The CompuScale Symbolic Name may be taken from CompuScale.symbol, CompuConstTextContent.vt, or CompuScale.shortLabel. The details are explained in [TPS_SWCT_01431].]()

[TPS_SWCT_01431] Finding the symbol for the representation of a CompuScale with a point-range in C code [In general, the value of the attributes symbol, vt, and shortLabel can be taken as a the source for naming the symbol that represents the CompuScale in the C code. The following rule applies (lower values indicate higher priority) for all CompuScales with a point-range:

1. Take the value of symbol if this attribute exists.
2. Take the value of vt if it makes a valid C identifier.

3. Take the value of `shortLabel` if it exists.

Fail if none of the possible options apply.

]()

[constr_1133] Identical CompuScale Symbolic Names shall have the same range [In a `CompuMethod` that is subject to [constr_1146], all `CompuScales` that yield identical `CompuScale Symbolic Names` shall have the same range defined by `CompuScale.lowerLimit` and `CompuScale.upperLimit`.]()

[constr_1146] Applicability of a symbol for a CompuScale in C code [The `symbol` attribute shall only be provided for `CompuScales` where the `category` of the enclosing `CompuMethod` is one of the following:

- `SCALE_LINEAR_AND_TEXTTABLE`
- `SCALE_RATIONAL_AND_TEXTTABLE`
- `TEXTTABLE`
- `BITFIELD_TEXTTABLE`

]()

Class	Compu			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::ComputationMethod			
Note	This meta-class represents the ability to express one particular computation.			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
compuContent	<code>CompuContent</code>	1	aggr	This specifies the details of the computation. Tags: xml.roleElement=false; xml.roleWrapperElement=false; xml.sequenceOffset=20; xml.typeElement=false; xml.typeWrapperElement=false
compuDefaultValue	<code>CompuConst</code>	0..1	aggr	This property can be used to specify an output value for a conversion formula, if the value to be converted lies outside the plausibility limit. Although this is possible for all conversion formulae, it is especially valid for variables with tabular conversion formulae. Tags: xml.sequenceOffset=70

Table 5.63: Compu

Class	CompuContent (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::ComputationMethod			
Note	This abstract meta-class represents the various definition means of a computation method.			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note

Attribute	Datatype	Mul.	Kind	Note
—	—	—	—	—

Table 5.64: CompuContent

Class	CompuScale			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::ComputationMethod			
Note	This meta-class represents the ability to specify one segment of a segmented computation method.			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
desc	MultiLanguage OverviewParagraph	0..1	aggr	<desc> represents a general but brief description of the object in question. Tags: xml.sequenceOffset=30
compulInverseValue	CompuConst	0..1	aggr	This is the inverse value of the constraint. This supports the case that the scale is not reversible per se. Tags: xml.sequenceOffset=60
compuScaleContents	CompuScaleContents	0..1	aggr	This represents the computation details of the scale. Tags: xml.roleElement=false; xml.roleWrapperElement=false; xml.sequenceOffset=70; xml.typeElement=false; xml.typeWrapperElement=false
lowerLimit	Limit	0..1	ref	This specifies the lower limit of the scale. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime xml.sequenceOffset=40
mask	PositiveInteger	0..1	attr	In difference to all the other computational methods every COMPU-SCALE will be applied including the bit MASK. Therefore it is allowed for this type of COMPU-METHOD, that COMPU-SCALES overlap. To calculate the string reverse to a value, the string has to be split and the according value for each substring has to be summed up. The sum is finally transmitted. The processing has to be done in order of the COMPU-SCALE elements. Tags: xml.sequenceOffset=35
shortLabel	Identifier	0..1	ref	This element specifies a short name for the particular scale. The name can for example be used to derive a programming language identifier. Tags: xml.sequenceOffset=20

Attribute	Datatype	Mul.	Kind	Note
symbol	CIdentifier	0..1	ref	The symbol, if provided, is used by code generators to get a C identifier for the CompuScale. The name will be used as is for the code generation, therefore it needs to be unique within the generation context. Tags: xml.sequenceOffset=25
upperLimit	Limit	0..1	ref	This specifies the upper limit of a of the scale. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime xml.sequenceOffset=50

Table 5.65: CompuScale

Class	CompuScales			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::ComputationMethod			
Note	This meta-class represents the ability to stepwise express a computation method.			
Base	ARObject, CompuContent			
Attribute	Datatype	Mul.	Kind	Note
compuScale (ordered)	CompuScale	*	aggr	This represents one scale within the compu method. Note that it contains a Variationpoint in order to support blueprints of enumerations. Stereotypes: atpVariation Tags: vh.latestBindingTime=blueprintDerivationTime xml.roleElement=true; xml.roleWrapperElement=true; xml.sequenceOffset=40; xml.typeElement=false; xml.typeWrapperElement=false

Table 5.66: CompuScales

Class	CompuScaleContents (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::ComputationMethod			
Note	This abstract meta-class represents the content of one particular scale.			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
—	—	—	—	—

Table 5.67: CompuScaleContents

Class	CompuRationalCoeffs			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::ComputationMethod			
Note	This meta-class represents the ability to express a rational function by specifying the coefficients of nominator and denominator.			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
compuDenominator	CompuNominatorDenominator	1	aggr	This is the denominator of the expression. Tags: xml.sequenceOffset=30
compuNumerator	CompuNominatorDenominator	1	aggr	This is the numerator of the rational expression. Tags: xml.sequenceOffset=20

Table 5.68: CompuRationalCoeffs

Class	CompuConst			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::ComputationMethod			
Note	This meta-class represents the fact that the value of a computation method scale is constant.			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
compuConstContentType	CompuConstContent	1	aggr	This is the actual content of the constant computation method scale. Tags: xml.roleElement=false; xml.roleWrapperElement=false; xml.sequenceOffset=10; xml.typeElement=false; xml.typeWrapperElement=false

Table 5.69: CompuConst

[TPS_SWCT_01429] [[constr_1135](#)] only applies for [BITFIELD_TEXTTABLE](#) [Note that [[constr_1135](#)] only applies for [BITFIELD_TEXTTABLE](#). It does **not** apply to the definition of [vt](#) in the context of an [ApplicationValueSpecification](#).]()

Class	CompuScaleRationalFormula			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::ComputationMethod			
Note	This meta-class represents the fact that the computation in this scale is represented as rational term.			
Base	ARObject, CompuScaleContents			
Attribute	Datatype	Mul.	Kind	Note
compuRationalCoeffs	CompuRationalCoeffs	1	aggr	This specifies the coefficients of the rational formula. Tags: xml.sequenceOffset=110

Table 5.70: CompuScaleRationalFormula

Class	CompuScaleConstantContents			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::ComputationMethod			
Note	This meta-class represents the fact that a particular scale of the computation method is constant.			
Base	ARObject, CompuScaleContents			
Attribute	Datatype	Mul.	Kind	Note
compuConstant	CompuConst	1	aggr	This represents the fact that the scale is a constant. The use case is mainly a non interpolated scale. It is a simplification of the fact that a constant scale can also be expressed as Rational Function of order 0. Tags: xml.sequenceOffset=90

Table 5.71: CompuScaleConstantContents

Class	CompuNominatorDenominator			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::ComputationMethod			
Note	This class represents the ability to express a polynomial either as Nominator or as Denominator.			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
v (ordered)	Numerical	*	attr	this is the list of polynomial factors. Note that the first vf represents the power=0. The polynomial is $v[0] * x^0 + v[1] * x^1 \dots$ Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime xml.roleElement=true; xml.roleWrapperElement=false; xml.sequenceOffset=20; xml.typeElement=false; xml.typeWrapperElement=false

Table 5.72: CompuNominatorDenominator

Please note that the values of coefficients within a rational formula are **not restricted** to integer values. It is possible to use floating point values as well.

The values of exponents **cannot be set arbitrarily** but are implicitly defined by the appearance of coefficients in [CompuNominatorDenominator.v](#), i.e. the first value in the ordered list of [CompuNominatorDenominator.v](#) represents the exponent 0, the second [CompuNominatorDenominator.v](#) represents the exponent 1, and so on.

5.5.1.1 Category Values in the context of a CompuMethod

For a detailed description of [CompuMethods](#), please refer to the *ASAM MCD 2 Harmonized Data Objects* [23].

Table 5.73 contains a definition of possible values for the attribute [category](#).

ASAM Category	Meaning	Specific properties
IDENTICAL	This <code>CompuMethod</code> just hands over the internal value with an optional unit.	Only the base elements are allowed and <code>unit</code> , <code>physConstr</code> and <code>internalConstr</code> are optional. This is the simplest type of a <code>CompuMethod</code> .
LINEAR	A linear conversion can be performed in two steps: The internal value is multiplied with a factor; after that, an offset is added to the result of the multiplication.	Exactly one <code>CompuScale</code> , with two <code>v</code> in <code>compuNumerator</code> and one <code>v</code> in <code>compuDenominator</code> .
SCALE_LINEAR	Used for a piecewise linear conversion	More than one <code>compuScale</code> can be defined. Additionally there have to be the <code>upperLimit</code> and <code>lowerLimit</code> elements which define the region of validity for the linear function. The boundaries of the regions shall not overlap.
SCALE_LINEAR_AND_TEXTTABLE	Used for piecewise definition of one linear and several texttable scales.	Properties depend on the used scale function. For details see definition of <code>SCALE_LINEAR</code> and <code>TEXTTABLE</code> . The scales shall each provide <code>lowerLimit</code> and <code>upperLimit</code> definitions.
RAT_FUNC	The rational function type is similar to the linear type without the restrictions for the <code>compuNumerators</code> and <code>compuDenominators</code> .	It can have as many <code>v</code> elements as needed for the rational function. The sequence of the values <code>v</code> carries the information for the exponents, that means the first <code>v</code> is the coefficient for <code>x0</code> , the second <code>v</code> is the coefficient for <code>x1</code> , etc. With this sequence the values of the exponents can be entirely represented. A rational function is only applicable for conversions in the direction that it is defined for, i.e. the automatic calculation of the inverse function is not supported by the MCD system.
SCALE_RAT_FUNC	Used for piecewise defined rational conversion.	
SCALE_RATIONAL_AND_TEXTTABLE	Used for piecewise definition of one rational and several texttable scales.	Properties depend on the used scale function. For details see definition of <code>SCALE_RAT_FUNC</code> and <code>TEXTTABLE</code> . The scales shall each provide <code>lowerLimit</code> and <code>upperLimit</code> definitions.
TEXTTABLE	The type <code>TEXTTABLE</code> is used for transformations of the internal value into textual elements.	[constr_1134] Allowed structure of <code>TEXTTABLE</code> [<code>physConstr</code> is not allowed. <code>compuInternalToPhys</code> shall exist with <code>compuScales</code> consisting of <code>upperLimit</code> and <code>lowerLimit</code> .]() The result is placed in the <code>vt</code> member of <code>CompuConst</code> . The <code>compuDefaultValue</code> is optional. If the reverse calculation is needed then for each scale the <code>compuInverseValue</code> can be used to define the reverse calculation result. If no inverse value is explicitly defined then the smallest possible value of the scale will be used as result of the reverse calculation.
TAB_NOINTP	Similar to <code>TEXTTABLE</code> , but for numerical values.	The values per scale are defined in <code>CompuConst</code> .

ASAM Category	Meaning	Specific properties
BITFIELD_TEXTTABLE	Similar to TEXTTABLE but for bit fields	BITFIELD_TEXTTABLE is derived from TEXTTABLE. The main difference is that TEXTTABLE results to a single value while BITFIELD_TEXTTABLE results to a concatenated value set. [constr_1135] Limit of vt in BITFIELD_TEXTTABLE [The separator is “ ” and is forbidden in vt therefore.]() In difference to all the other computational methods every CompuScale will be applied including the bit mask specified in mask. Therefore it is allowed for this type of CompuMethod, that CompuScales overlap. To calculate the string reverse to a value, the string has to be split and the according value for each substring has to be summed up. The sum is finally transmitted. The processing has to be done in order of the CompuScale elements.

Table 5.73: ASAM compuMethod

5.5.1.2 Applicability of Attributes in the context of a CompuMethod

This section summarizes the applicability of [CompuMethod](#) in terms of which attributes of [CompuMethod](#) and related meta-classes (e.g. [CompuScale](#), [CompuConst](#)) shall be used depending on the nature of the [CompuMethod](#), expressed by means of the value of attribute [category](#).

[constr_1375] Existence of attributes of [CompuMethod](#) and related meta-classes
[The existence of attributes of [CompuMethod](#) and related meta-classes depending on the value of the [category](#) shall follow the restrictions documented in Table 5.74.]()

For clarification, the first two rows of Table 5.74 define the applicability of the immediate attributes of meta-class [CompuMethod](#), the remainder of the table then goes into further detail regarding the usage of the attributes of related meta-classes (e.g. [CompuScale](#), [CompuConst](#)).

Please note that annotations apply to the individual cell values. These annotations are formulated by means of a numerical value in parentheses, e.g. (1).

The legend for the individual annotations can be found below Table 5.74.

	Attribute Existence per Category									
	IDENTICAL	LINEAR	SCALE_LINEAR	RAT_FUNC	SCALE_RAT_FUNC	TEXTTABLE	BITFIELD_TEXTTABLE	SCALE_LINEAR_AND_TEXTTABLE	SCALE_RATIONAL_AND_TEXTTABLE	TAB_NOINTP
Attributes of CompuMethod										
<code>compuInternalToPhys</code>	N/A	D(1)	D(1)	D(2)	D(2)	D	D	D(8)	D(2)	D
<code>compuPhysToInternal</code>	N/A	D	D	D(2)	D(2)	N/A	N/A	N/A	D(2,3)	N/A
Attributes of meta-classes related to CompuMethod										
<code>compuDefaultValue</code>	N/A	O(6)	O(6)	O(6)	O(6)	O(6)	O(6)	O(6)	O(6)	O(6)
<code>CompuScale</code>	N/A	D/1..1	D/1..n	D/1..1	D/1..n	D/1..n	D/1..n	D/1..n	D/1..n	D/1..n
<code>CompuScale.compuInverseValue</code>	N/A	N/A	N/A	O(2)	O(2)	O(5)	N/A	O(2,5)	O(2,5)	O(5)
<code>CompuScale.lowerLimit</code>	N/A	O	D	D(4)	D(4)	D	D	D	D(4)	D
<code>CompuScale.mask</code>	N/A	N/A	N/A	N/A	N/A	N/A	D	N/A	N/A	N/A
<code>CompuScale.shortLabel</code>	N/A	N/A	N/A	N/A	N/A	O(7)	O(7)	O(7)	O(7)	N/A
<code>CompuScale.symbol</code>	N/A	N/A	N/A	N/A	N/A	O(7)	O(7)	O(7)	O(7)	N/A
<code>CompuScale.upperLimit</code>	N/A	O	D	D(4)	D(4)	D	D	D	D(4)	D
<code>CompuConst</code>	N/A	N/A	N/A	N/A	N/A	D/vt	D/vt	D/vt	D/vt	D/vt or vf
<code>CompuRationalCoeffs</code>	N/A	D	D	D	D	N/A	N/A	D	D	N/A
<code>CompuRationalCoeffs.compuDenominator</code>	N/A	D/1v	D/1v	D	D	N/A	N/A	D/1v	D	N/A
<code>CompuRationalCoeffs.compuDenominator</code>	N/A	D/2v	D/2v	D	D	N/A	N/A	D/2v	D	N/A

Table 5.74: Allowed Attributes vs. category for CompuMethods

The following legend applies to the cells in table 5.74:

D Define the attribute.

N/A Attribute is **not applicable** for usage in the scope of this element.

O Optionally define the attribute.

In addition to the primary cell legend the following annotations apply to the cells in table 5.74:

- (1) This applies if not already defined by `compuPhysToInternal`.
- (2) In this case both `compuPhysToInternal` and `compuInternalToPhys` shall be defined (according to [constr_1021]) unless `compuInverseValue` exists (see [TPS_SWCT_01282]). In other words, if the explicit definition of a `compuInverseValue` exists then there is no need to define conversions from internal to physical **and** vice versa.
- (3) Not applicable for `CompuScales` where attribute `compuScaleContents.compuConst` exists.
- (4) Limits shall be defined according to [constr_1022].

- (5) Restrictions on the structure of the [CompuMethod](#) according to [[constr_1134](#)] apply.
- (6) Specify an output value for a conversion formula if the value to be converted yields outside the plausibility limit (for more information, please refer to the class table of [Compu](#)).
- (7) Restricted applicability for the attribute [CompuScale.symbol](#), see [[constr_1146](#)]).
- (8) Mandatory for [CompuConst](#); enforced for [CompuRationalCoeffs](#).

5.5.1.3 Example for Enumeration

The following example illustrates how an enumeration is specified using [CompuMethod](#).

Listing 5.7: example for enumeration

```
<COMPU-METHOD>
  <SHORT-NAME>boolean</SHORT-NAME>
  <CATEGORY>TEXTTABLE</CATEGORY>
  <COMPU-INTERNAL-TO-PHYS>
    <COMPU-SCALES>
      <COMPU-SCALE>
        <LOWER-LIMIT INTERVAL-TYPE="CLOSED">0</LOWER-LIMIT>
        <UPPER-LIMIT INTERVAL-TYPE="CLOSED">0</UPPER-LIMIT>
        <COMPU-CONST>
          <VT>false</VT>
        </COMPU-CONST>
      </COMPU-SCALE>
      <COMPU-SCALE>
        <LOWER-LIMIT INTERVAL-TYPE="CLOSED">1</LOWER-LIMIT>
        <UPPER-LIMIT INTERVAL-TYPE="CLOSED">1</UPPER-LIMIT>
        <COMPU-CONST>
          <VT>true</VT>
        </COMPU-CONST>
      </COMPU-SCALE>
    </COMPU-SCALES>
  </COMPU-INTERNAL-TO-PHYS>
</COMPU-METHOD>
```

5.5.1.4 Example for Linear Conversion

The following examples illustrates how a linear conversion is specified using [CompuMethod](#).

$$F_{[kmh]} = 30_{[kmh]} + 2_{[kmh]} * x$$

Listing 5.8: example for linear [CompuMethod](#)

```
<COMPU-METHOD>
  <SHORT-NAME>linear</SHORT-NAME>
```

```

<CATEGORY>LINEAR</CATEGORY>
<UNIT-REF DEST="UNIT">kmh</UNIT-REF>
<COMPU-INTERNAL-TO-PHYS>
  <COMPU-SCALES>
    <COMPU-SCALE>
      <COMPU-RATIONAL-COEFFS>
        <COMPU-NUMERATOR>
          <V>30</V>
          <V>2</V>
        </COMPU-NUMERATOR>
        <COMPU-DENOMINATOR>
          <V>1</V>
        </COMPU-DENOMINATOR>
      </COMPU-RATIONAL-COEFFS>
    </COMPU-SCALE>
  </COMPU-SCALES>
</COMPU-INTERNAL-TO-PHYS>
</COMPU-METHOD>

```

5.5.1.5 Example for Linear Conversion with texttable

The following example illustrates how a linear conversion with a texttable is specified using [CompuMethod](#).

Listing 5.9: example for linear and texttable [CompuMethod](#)

```

<COMPU-METHOD>
  <SHORT-NAME>linear</SHORT-NAME>
  <CATEGORY>SCALE_LINEAR_AND_TEXTTABLE</CATEGORY>
  <UNIT-REF DEST="UNIT">kmh</UNIT-REF>
  <COMPU-INTERNAL-TO-PHYS>
    <COMPU-SCALES>
      <COMPU-SCALE>
        <LOWER-LIMIT INTERVAL-TYPE="CLOSED">0</LOWER-LIMIT>
        <UPPER-LIMIT INTERVAL-TYPE="CLOSED">300</UPPER-LIMIT>
        <COMPU-RATIONAL-COEFFS>
          <COMPU-NUMERATOR>
            <V>30</V>
            <V>2</V>
          </COMPU-NUMERATOR>
          <COMPU-DENOMINATOR>
            <V>1</V>
          </COMPU-DENOMINATOR>
        </COMPU-RATIONAL-COEFFS>
      </COMPU-SCALE>
      <COMPU-SCALE>
        <LOWER-LIMIT INTERVAL-TYPE="CLOSED">350</LOWER-LIMIT>
        <UPPER-LIMIT INTERVAL-TYPE="CLOSED">350</UPPER-LIMIT>
        <COMPU-CONST>
          <VT>SensorError</VT>
        </COMPU-CONST>
      </COMPU-SCALE>
      <COMPU-SCALE>
        <LOWER-LIMIT INTERVAL-TYPE="CLOSED">351</LOWER-LIMIT>

```



```

    <UPPER-LIMIT INTERVAL-TYPE="CLOSED">351</UPPER-LIMIT>
    <COMPU-CONST>
      <VT>SignalNotAvailable</VT>
    </COMPU-CONST>
  </COMPU-SCALE>
</COMPU-SCALES>
</COMPU-INTERNAL-TO-PHYS>
</COMPU-METHOD>

```

5.5.1.6 Example for conversion specified by a rational function

The semantics of rational function is:

$$Internal = \frac{v_0 * phys^0 + v_1 * phys^1 + v_2 * phys^2 + \dots}{v_0 * phys^0 + v_1 * phys^1 + v_2 * phys^2 + \dots}$$

The following example illustrates a reciprocal conversion.

$$I = \frac{1000}{60 + 2_{[K-1]} * P_{[K]}}$$

Listing 5.10: example for rational [CompuMethod](#)

```

<COMPU-METHOD>
  <SHORT-NAME>rational</SHORT-NAME>
  <CATEGORY>RAT_FUNC</CATEGORY>
  <UNIT-REF DEST="UNIT">Kelvin</UNIT-REF>
  <COMPU-PHYS-TO-INTERNAL>
    <COMPU-SCALES>
      <COMPU-SCALE>
        <LOWER-LIMIT INTERVAL-TYPE="CLOSED">-29</LOWER-LIMIT>
        <UPPER-LIMIT INTERVAL-TYPE="OPEN">INF</UPPER-LIMIT>
        <COMPU-RATIONAL-COEFFS>
          <COMPU-NUMERATOR>
            <V>1000</V>
          </COMPU-NUMERATOR>
          <COMPU-DENOMINATOR>
            <V>60</V>
            <V>2</V>
          </COMPU-DENOMINATOR>
        </COMPU-RATIONAL-COEFFS>
      </COMPU-SCALE>
    </COMPU-SCALES>
  </COMPU-PHYS-TO-INTERNAL>
</COMPU-METHOD>

```

5.5.1.7 Example for BITFIELD_TEXTTABLE

The following example shows how a [CompuMethod](#) of category [BITFIELD_TEXTTABLE](#) can be used to assign a special meaning to each bit of an [AutosarDataType](#) of category [VALUE](#):

Bit 0	front left	0(0) = no, 1(1) = yes

Bit 1	front right	0(0) = no, 1(2) = yes
Bit 2	rear left	0(0) = no, 1(4) = yes
Bit 3	rear right	0(0) = no, 1(8) = yes
Bit 4-5	problem	00(0) = flat tire 01(16) = low pressure 10(32) = unbalanced 11(48) = unknown
All Bits	error	11111111 = invalid value

Table 5.75: Example Bitfield

Note that this example is somehow tricky. Bit 6+7 are not used for valid data, but are part of the mask. By this the error can safely be masked out.

Internal: 28

28 = 0b0001_1100
Bit 7654 3210

Physical:

"problem = low pressure | rear right = yes | rear left = yes | front right = no | front left = no"

Listing 5.11: example for bit field text table [CompuMethod](#)

```

<COMPU-METHOD>
  <SHORT-NAME>Texttable</SHORT-NAME>
  <CATEGORY>BITFIELD_TEXTTABLE</CATEGORY>
  <COMPU-INTERNAL-TO-PHYS>
    <COMPU-SCALES>
      <!-- problem -->
      <COMPU-SCALE>
        <SHORT-LABEL>problem</SHORT-LABEL>
        <SYMBOL>problem_flat_tire</SYMBOL>
        <MASK>0b11110000</MASK>
        <LOWER-LIMIT INTERVAL-TYPE="CLOSED">0b00000000</LOWER-LIMIT>
        <UPPER-LIMIT INTERVAL-TYPE="CLOSED">0b00000000</UPPER-LIMIT>
        <COMPU-CONST>
          <VT>flat tire</VT>
        </COMPU-CONST>
      </COMPU-SCALE>
      <COMPU-SCALE>
        <SHORT-LABEL>problem</SHORT-LABEL>
        <SYMBOL>problem_low_pressure</SYMBOL>
        <MASK>0b11110000</MASK>
        <LOWER-LIMIT INTERVAL-TYPE="CLOSED">0b00010000</LOWER-LIMIT>
        <UPPER-LIMIT INTERVAL-TYPE="CLOSED">0b00010000</UPPER-LIMIT>
        <COMPU-CONST>
          <VT>low pressure</VT>
        </COMPU-CONST>
      </COMPU-SCALE>
      <COMPU-SCALE>
        <SHORT-LABEL>problem</SHORT-LABEL>
        <SYMBOL>problem_unbalanced</SYMBOL>
        <MASK>0b11110000</MASK>
        <LOWER-LIMIT INTERVAL-TYPE="CLOSED">0b00100000</LOWER-LIMIT>

```

```

    <UPPER-LIMIT INTERVAL-TYPE="CLOSED">0b00100000</UPPER-LIMIT>
    <COMPU-CONST>
      <VT>unbalanced</VT>
    </COMPU-CONST>
  </COMPU-SCALE>
</COMPU-SCALE>
<COMPU-SCALE>
  <SHORT-LABEL>problem</SHORT-LABEL>
  <SYMBOL>problem_unknown</SYMBOL>
  <MASK>0b11110000</MASK>
  <LOWER-LIMIT INTERVAL-TYPE="CLOSED">0b00110000</LOWER-LIMIT>
  <UPPER-LIMIT INTERVAL-TYPE="CLOSED">0b00110000</UPPER-LIMIT>
  <COMPU-CONST>
    <VT>unknown</VT>
  </COMPU-CONST>
</COMPU-SCALE>
</COMPU-SCALE>
<COMPU-SCALE>
  <SHORT-LABEL>problem</SHORT-LABEL>
  <SYMBOL>problem_invalid</SYMBOL>
  <MASK>0b11110000</MASK>
  <LOWER-LIMIT INTERVAL-TYPE="CLOSED">0b11110000</LOWER-LIMIT>
  <UPPER-LIMIT INTERVAL-TYPE="CLOSED">0b11110000</UPPER-LIMIT>
  <COMPU-CONST>
    <VT>invalid</VT>
  </COMPU-CONST>
</COMPU-SCALE>
<!-- rear right -->
<COMPU-SCALE>
  <SHORT-LABEL>rearRight</SHORT-LABEL>
  <SYMBOL>rearRight_no</SYMBOL>
  <MASK>0b11001000</MASK>
  <LOWER-LIMIT INTERVAL-TYPE="CLOSED">0b00000000</LOWER-LIMIT>
  <UPPER-LIMIT INTERVAL-TYPE="CLOSED">0b00000000</UPPER-LIMIT>
  <COMPU-CONST>
    <VT>no</VT>
  </COMPU-CONST>
</COMPU-SCALE>
</COMPU-SCALE>
<COMPU-SCALE>
  <SHORT-LABEL>rearRight</SHORT-LABEL>
  <SYMBOL>rearRight_yes</SYMBOL>
  <MASK>0b11001000</MASK>
  <LOWER-LIMIT INTERVAL-TYPE="CLOSED">0b00001000</LOWER-LIMIT>
  <UPPER-LIMIT INTERVAL-TYPE="CLOSED">0b00001000</UPPER-LIMIT>
  <COMPU-CONST>
    <VT>yes</VT>
  </COMPU-CONST>
</COMPU-SCALE>
<!-- rear left -->
<COMPU-SCALE>
  <SHORT-LABEL>rearLeft</SHORT-LABEL>
  <SYMBOL>rearLeft_no</SYMBOL>
  <MASK>0b11000100</MASK>
  <LOWER-LIMIT INTERVAL-TYPE="CLOSED">0b00000000</LOWER-LIMIT>
  <UPPER-LIMIT INTERVAL-TYPE="CLOSED">0b00000000</UPPER-LIMIT>
  <COMPU-CONST>
    <VT>no</VT>
  </COMPU-CONST>
</COMPU-SCALE>

```

```

</COMPU-SCALE>
<COMPU-SCALE>
  <SHORT-LABEL>rearLeft</SHORT-LABEL>
  <SYMBOL>rearLeft_yes</SYMBOL>
  <MASK>0b11000100</MASK>
  <LOWER-LIMIT INTERVAL-TYPE="CLOSED">0b00000100</LOWER-LIMIT>
  <UPPER-LIMIT INTERVAL-TYPE="CLOSED">0b00000100</UPPER-LIMIT>
  <COMPU-CONST>
    <VT>yes</VT>
  </COMPU-CONST>
</COMPU-SCALE>
<!-- front right -->
<COMPU-SCALE>
  <SHORT-LABEL>frontRight</SHORT-LABEL>
  <SYMBOL>frontRight_no</SYMBOL>
  <MASK>0b11000010</MASK>
  <LOWER-LIMIT INTERVAL-TYPE="CLOSED">0b00000000</LOWER-LIMIT>
  <UPPER-LIMIT INTERVAL-TYPE="CLOSED">0b00000000</UPPER-LIMIT>
  <COMPU-CONST>
    <VT>no</VT>
  </COMPU-CONST>
</COMPU-SCALE>
<COMPU-SCALE>
  <SHORT-LABEL>frontRight</SHORT-LABEL>
  <SYMBOL>frontRight_yes</SYMBOL>
  <MASK>0b11000010</MASK>
  <LOWER-LIMIT INTERVAL-TYPE="CLOSED">0b00000010</LOWER-LIMIT>
  <UPPER-LIMIT INTERVAL-TYPE="CLOSED">0b00000010</UPPER-LIMIT>
  <COMPU-CONST>
    <VT>yes</VT>
  </COMPU-CONST>
</COMPU-SCALE>
<!-- front left -->
<COMPU-SCALE>
  <SHORT-LABEL>frontLeft</SHORT-LABEL>
  <SYMBOL>frontLeft_no</SYMBOL>
  <MASK>0b11000001</MASK>
  <LOWER-LIMIT INTERVAL-TYPE="CLOSED">0b00000000</LOWER-LIMIT>
  <UPPER-LIMIT INTERVAL-TYPE="CLOSED">0b00000000</UPPER-LIMIT>
  <COMPU-CONST>
    <VT>no</VT>
  </COMPU-CONST>
</COMPU-SCALE>
<COMPU-SCALE>
  <SHORT-LABEL>frontLeft</SHORT-LABEL>
  <SYMBOL>frontLeft_yes</SYMBOL>
  <MASK>0b11000001</MASK>
  <LOWER-LIMIT INTERVAL-TYPE="CLOSED">0b00000001</LOWER-LIMIT>
  <UPPER-LIMIT INTERVAL-TYPE="CLOSED">0b00000001</UPPER-LIMIT>
  <COMPU-CONST>
    <VT>yes</VT>
  </COMPU-CONST>
</COMPU-SCALE>
</COMPU-SCALES>
</COMPU-INTERNAL-TO-PHYS>
</COMPU-METHOD>

```

Note that a constraint applies concerning the values within a `CompuMethod` that is subject to [constr_1146]. According to [constr_1133], it is (as exemplified in the example) required that if a specific `CompuScale` Symbolic Name appears more than once in the context of the `CompuMethod` then the values of `CompuScale.lowerLimit` shall be identical for all affected `CompuScales` and the values of `CompuScale.upperLimit` shall also be identical for all affected `CompuScales`.

Class	CompuScaleContents (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::ComputationMethod			
Note	This abstract meta-class represents the content of one particular scale.			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
–	–	–	–	–

Table 5.76: CompuScaleContents

Class	CompuConstTextContent			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::ComputationMethod			
Note	This meta-class represents the textual content of a scale.			
Base	ARObject,CompuConstContent			
Attribute	Datatype	Mul.	Kind	Note
vt	VerbatimString	1	ref	This represents a textual constant in the computation method.

Table 5.77: CompuConstTextContent

Class	CompuConstNumericContent			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::ComputationMethod			
Note	This meta-class represents the fact that the constant value of the computation method is a numerical value. It is separated from <code>CompuConstFormulaContent</code> to support compatibility with ASAM HDO.			
Base	ARObject,CompuConstContent			
Attribute	Datatype	Mul.	Kind	Note
v	Numerical	1	attr	This represents the numerical value. Tags: xml.sequenceOffset=50

Table 5.78: CompuConstNumericContent

5.5.2 Physical Units, Physical Dimensions and Unit Groups

[TPS_SWCT_01285] Physical dimension [Another important part of the semantics associated with a data type is its physical dimension. Units are used to augment the value with additional information like *m/s* or *liter*. This is necessary for a correct interpretation of the physical value for input and output processes.

The conversion of values into other units like *km/h* into *miles/h* is also possible. Therefore the unit involves information about its physical dimensions. $\rfloor()$

[TPS_SWCT_01056] Physical dimension $\rfloor$ The substructure of physical dimensions defines all used quantities in the SI-System¹² (e.g. velocity as length/time corresponds to m/s). $\rfloor$ ([RS_SWCT_02100](#))

[TPS_SWCT_01057] Unit references one physical dimension $\rfloor$ The unit references one physical dimension. If the physical dimensions of two units are identical, a conversion between them is basically possible. $\rfloor$ ([RS_SWCT_02100](#))

[TPS_SWCT_01058] UnitGroup $\rfloor$ The `UnitGroups` determine if such a conversion is appropriate. $\rfloor$ ([RS_SWCT_02100](#))

Figure 5.38 depicts the concept how units are defined.

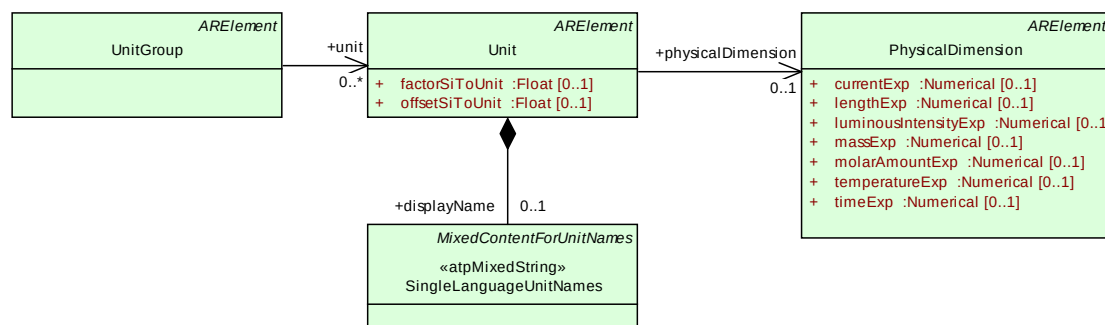


Figure 5.38: Definition of SI based units

For a detailed description of these elements please refer to the [23]. Standard units are already predefined for AUTOSAR in form of a description file.

<i>Class</i>	Unit			
<i>Package</i>	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::Units			
<i>Note</i>	This is a physical measurement unit. All units that might be defined should stem from SI units. In order to convert one unit into another factor and offset are defined. For the calculation from SI-unit to the defined unit the factor (factorSiToUnit) and the offset (offsetSiToUnit) are applied: $\text{unit} = \text{siUnit} * \text{factorSiToUnit} + \text{offsetSiToUnit}$ For the calculation from a unit to SI-unit the reciprocal of the factor (factorSiToUnit) and the negation of the offset (offsetSiToUnit) are applied: $\text{siUnit} = (\text{unit} - \text{offsetSiToUnit}) / \text{factorSiToUnit}$ Tags: atp.recommendedPackage=Units			
<i>Base</i>	ARElement ,ARObject,CollectableElement, Identifiable ,Multilanguage Referrable,PackageableElement, Referrable			
<i>Attribute</i>	<i>Datatype</i>	<i>Mul.</i>	<i>Kind</i>	<i>Note</i>

¹²For the definition of what SI units are, see <http://physics.nist.gov/cuu/Units/>

Attribute	Datatype	Mul.	Kind	Note
displayName	SingleLanguageUnitNames	0..1	aggr	This specifies how the unit shall be displayed in documents or in user interfaces of tools. The displayName corresponds to the Unit.Display in an ASAM MCD-2MC file. Tags: xml.sequenceOffset=20
factorSiToUnit	Float	0..1	attr	This is the factor for the conversion from and to siUnits. Tags: xml.sequenceOffset=30
offsetSiToUnit	Float	0..1	attr	This is the offset for the conversion from and to siUnits. Tags: xml.sequenceOffset=40
physicalDimension	PhysicalDimension	0..1	ref	This association represents the physical dimension to which the unit belongs to. Note that only values with units of the same physical dimensions might be converted. Tags: xml.sequenceOffset=50

Table 5.79: Unit

[TPS_SWCT_01059] Exponent for each of the seven fundamental dimensions

┌ For basing a new unit directly upon SI units an exponent for each of the seven fundamental dimensions and its corresponding SI unit needs to be specified.
└ ([RS_SWCT_02100](#))

[TPS_SWCT_01060] Negative exponents ┌ Negative exponents are allowed.

└ ([RS_SWCT_02100](#))

Note that quantities like "%" are not derived from SI units and therefore have no association to a physical dimension.

Class	PhysicalDimension			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::Units			
Note	This class represents a physical dimension. If the physical dimension of two units is identical, then a conversion between them is possible. The conversion between units is related to the definition of the physical dimension. Note that the equivalence of the exponents does not per se define the convertibility. For example Energy and Torque share the same exponents (Nm). Please note further the value of an exponent does not necessarily have to be an integer number. It is also possible that the value yields a rational number, e.g. to compute the square root of a given physical quantity. In this case the exponent value would be a rational number where the numerator value is 1 and the denominator value is 2. Tags: atp.recommendedPackage=PhysicalDimensions			
Base	ARElement , ARObject , CollectableElement , Identifiable , Multilanguage Referrable , PackageableElement , Referrable			
Attribute	Datatype	Mul.	Kind	Note
currentExp	Numerical	0..1	attr	This attribute represents the exponent of the physical dimension "electric current". Tags: xml.sequenceOffset=50
lengthExp	Numerical	0..1	attr	The exponent of the physical dimension "length". Tags: xml.sequenceOffset=20
luminousIntensityExp	Numerical	0..1	attr	The exponent of the physical dimension "luminous intensity". Tags: xml.sequenceOffset=80
massExp	Numerical	0..1	attr	The exponent of the physical dimension "mass". Tags: xml.sequenceOffset=30
molarAmountExp	Numerical	0..1	attr	The exponent of the physical dimension "quantity of substance". Tags: xml.sequenceOffset=70
temperatureExp	Numerical	0..1	attr	The exponent of the physical dimension "temperature". Tags: xml.sequenceOffset=60
timeExp	Numerical	0..1	attr	The exponent of the physical dimension "time". Tags: xml.sequenceOffset=40

Table 5.80: PhysicalDimension

AUTOSAR provides the ability to map two [PhysicalDimensions](#) onto each others with the implication that the two mapped [PhysicalDimensions](#) shall be considered compatible (for more explanation please refer to [\[constr_1053\]](#)). [PhysicalDimensionMappings](#) are aggregated in form of [PhysicalDimensionMappingSets](#). This

allows for gathering semantically related [PhysicalDimensionMappings](#) into the same [PhysicalDimensionMappingSet](#).

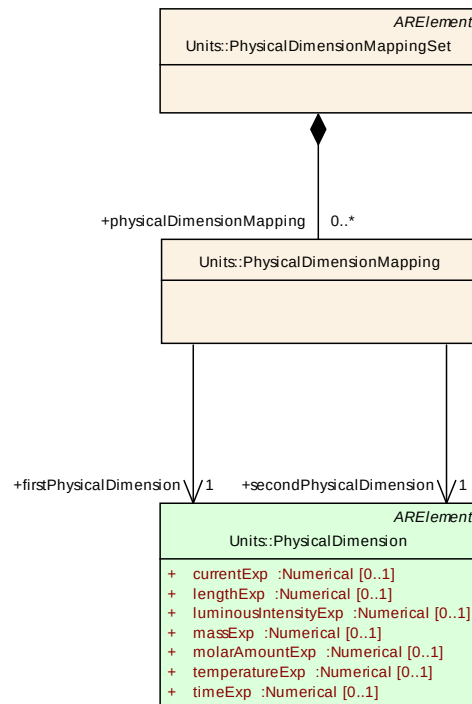


Figure 5.39: Modeling of [PhysicalDimensionMapping](#)

Class	PhysicalDimensionMappingSet			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::Units			
Note	This class represents a container for a list of mappings between PhysicalDimensions. Tags: atp.recommendedPackage=PhysicalDimensionMappingSets			
Base	ARElement , ARObject , CollectableElement , Identifiable , Multilanguage , Referrable , PackageableElement , Referrable			
Attribute	Datatype	Mul.	Kind	Note
physicalDimensionMapping	PhysicalDimensionMapping	*	aggr	This aggregation represents a concrete collections of PhysicalDimensionMappings in the context of one PhysicalDimensionMappingSet.

Table 5.81: PhysicalDimensionMappingSet

Class	PhysicalDimensionMapping			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::Units			
Note	This class represents a specific mapping between two PhysicalDimensions.			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
firstPhysicalDimension	PhysicalDimension	1	ref	This represents the first PhysicalDimension of the enclosing PhysicalDimensionMapping.

Attribute	Datatype	Mul.	Kind	Note
secondPhysicalDimension	PhysicalDimension	1	ref	This represents the first PhysicalDimension of the enclosing PhysicalDimensionMapping.

Table 5.82: PhysicalDimensionMapping

[constr_1026] Compatibility of Units [For data types or prototypes, units should be referenced from within the associated `CompuMethod`. But if it is referenced from within `SwDataDefProps` and/or `PhysConstrs` (for exceptional use cases) it shall be compatible (for more details please refer to [constr_1052]) to the ones referenced from the referred `CompuMethod`.]()

Please note that for the sake of model consistency, it is also possible to define a meaningless `Unit` for all the pieces of data that conceptually do not really have a `Unit` attached to them (e.g. `ApplicationPrimitiveDataTypes` of category `BOOLEAN`).

By looking at the model, it becomes clear that the subject of whether or not to assign a `Unit` has been given a thought and the lack of a `Unit` is not simply the result of an oversight. For example, the AUTOSAR Application Interfaces [24] define the `Unit NoUnit` for exactly this purpose.

[constr_1255] ApplicationPrimitiveDataTypes of category BOOLEAN and STRING [If a `Unit` is referenced from within `SwDataDefProps` and/or `PhysConstrs` owned by an `ApplicationPrimitiveDataTypes` of category `BOOLEAN` and `STRING` it is required that this `Unit` represents a meaningless unit, i.e. the referenced `physicalDimension` shall not define any exponent value other than 0.]()

Class	UnitGroup			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::Units			
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,PackageableElement,Referrable			
Attribute	Datatype	Mul.	Kind	Note

Attribute	Datatype	Mul.	Kind	Note
unit	Unit	*	ref	This represents one particular unit in the UnitGroup. Tags: xml.sequenceOffset=20

Table 5.83: UnitGroup

[TPS_SWCT_01068] **Units** can be grouped with the help of **UnitGroup** [Units can be grouped with the help of **UnitGroup**. This grouping is intended as a logical grouping which allows for example an MCD (Measurement Calibration Diagnostic) device to present different unit systems to the user such that he can chose the most appropriate one.] (RS_SWCT_02100)

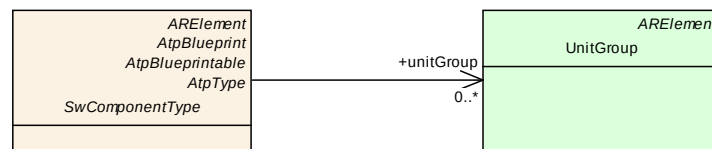


Figure 5.40: Relation of SwComponentType to UnitGroup

The association from **SwComponentType** to **UnitGroup** (beside the obvious use case to allow for the specification of **unitGroups** relevant for the enclosing **SwComponentType** in particular) is supposed to support the identification of **UnitGroups** relevant for the enclosing **System**. This aspect facilitates the creation of ASAM MCD2 files for a concrete ECU.

According to [23] the following three values for **categorys** are recommended in the context of **UnitGroup**:

- **COUNTRY** collects units which are common in a particular country, denoted by the **shortName** / **longName** of the **UnitGroup**
- **CALCULATION** refers to specific units intended for the creation of data types. In this **category** of **UnitGroup**, several **Units** may refer to the same **PhysicalDimension** as well as to different **PhysicalDimension**.
- **EQUIV_UNITS** define a group of equivalent units, which are used for example in different countries.

Additional values for **category** may be mutually agreed between the stakeholders.

In the example shown in Figure 5.41, **Units** are classified by country and use.

[TPS_SWCT_01061] **Conversion of units** [If a unit has to be converted according to the chosen country code the **physicalDimension** of both units shall be the same. If another unit shares the same **UnitGroup** with a **category** of **EQUIV_UNITS** it is preferred as target of the conversion.] (RS_SWCT_02100)

Assume "MilesPerHour" should be converted to a European unit: Based on the **physicalDimension** a conversion to "MeterPerSec" as well as "MilesPerHour" is possible. In this case "KmPerHour" is preferred because "MilesPerHour" and "KmPerHour" are

both members of the `UnitGroup` named "VehicleSpeed". In contrast to this "MeterPerSec" is not considered as appropriate for "VehicleSpeed".

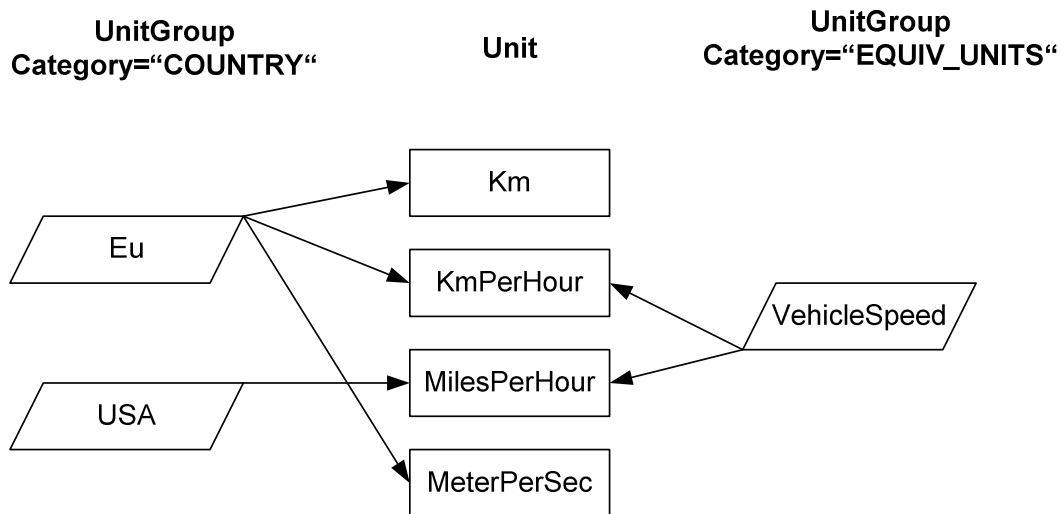


Figure 5.41: Example for units and unit groups

5.5.3 Data Constraints

Section 5.2.4.1 already shows an example on how to define constraints for the physical range of a data type, see Figure 5.4.

[TPS_SWCT_01286] `DataConstr` [In general, the meta-class `DataConstr` can be aggregated (via `SwDataDefProps.dataConstr`) to define various constraints for the possible values of a data type. This includes limits for the physical and internal range, as well as special constraints (`monotony`) for the setup of axis definition.]()

Figure 5.42 and the following class tables show the meta-classes involved in the definition of constraints.

A more detailed documentation of these meta-classes can be found in in [23]. As refinement of these definitions, the following values apply for `constrLevel`:

[constr_2561] Application of `DataConstrRule.constrLevel` [`DataConstrRule.constrLevel` is limited to

- 0:** This represents so called "hard limits". They shall always be specified.
- 1:** This represents so called "soft limits". Soft limits may be violated after confirmation by the user of an MCD-System.

Other values may exist, but the semantics is outside of the AUTOSAR scope.

]()

[TPS_SWCT_01287] Standard limits and extended limits in the ASAM-MCD2 (ASAP2) specification [The ASAM-MCD2 (ASAP2) specification [25] defines stan-

standard limits and extended limits. If extended limits exist, the standard limits may be violated upon user confirmation. Note that in consequence, of this definition, the following approach applies for A2L generation:

- If only one `DataConstrRule` with `constrLevel` set to **0** is specified, it represents the standard limits in A2L. No extended limits are generated.
- If two `DataConstrRule` exist, then:
 - the one with `constrLevel` set to **0** represents to the extended limits
 - the one with `constrLevel` set to **1** represents to the standard limits

Note that even if this is somehow counterintuitive (since the one with `constrLevel` set to 0 changes its role), it matches the best to the definitions in ASAM-MCD2. $\square()$

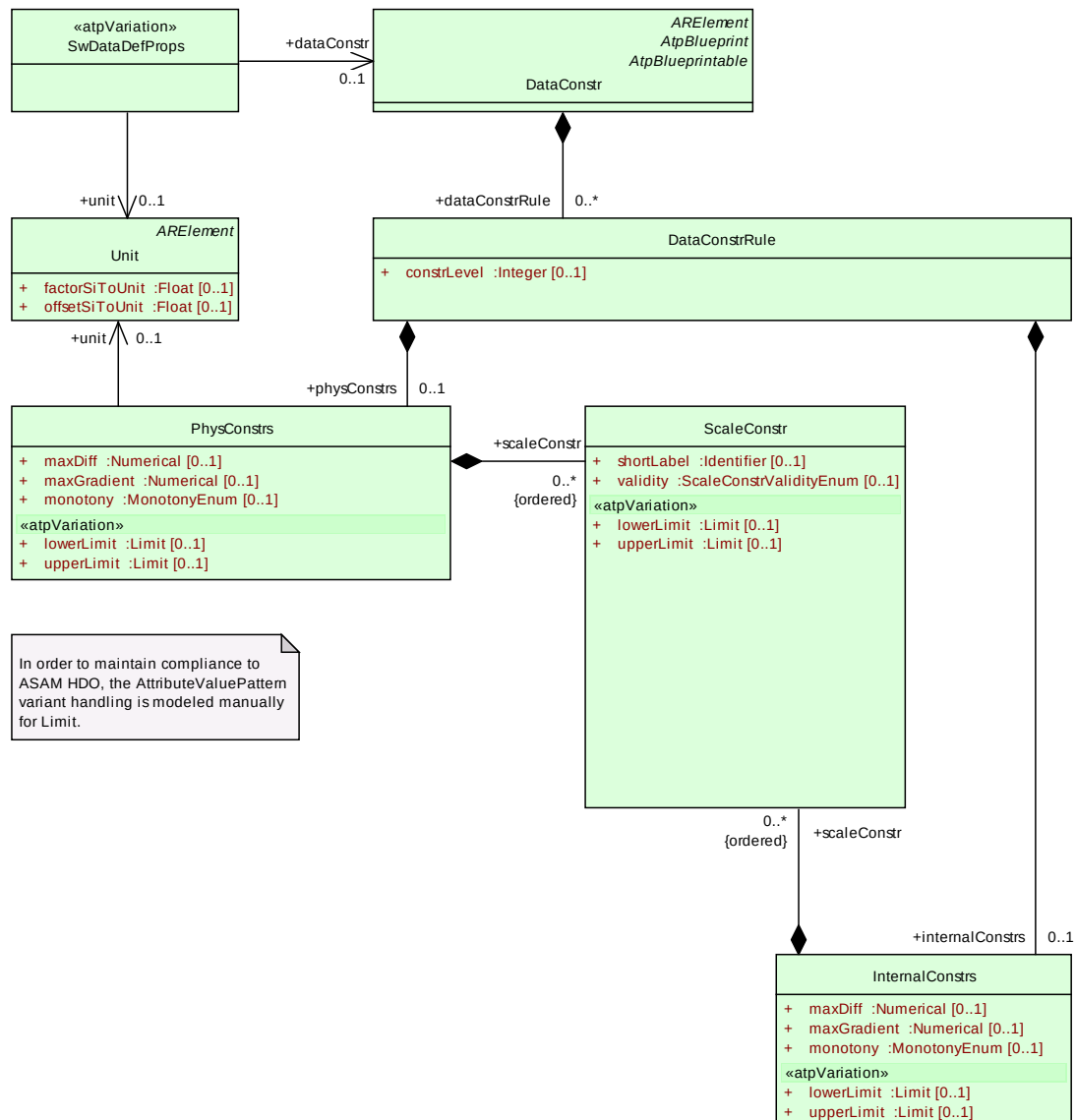


Figure 5.42: Meta-model for defining Data Constraints

Class	DataConstr			
Package	M2::AUTOSARTemplates::CommonStructure::GlobalConstraints			
Note	This meta-class represents the ability to specify constraints on data. Tags: atp.recommendedPackage=DataConstrs			
Base	ARElement , ARObject , AtpBlueprint , AtpBlueprintable , CollectableElement , Identifiable , MultilanguageReferrable , PackageableElement , Referrable			
Attribute	Datatype	Mul.	Kind	Note
dataConstrRule	DataConstrRule	*	aggr	This is one particular rule within the data constraints. Tags: xml.roleElement=true; xml.roleWrapperElement=true; xml.sequenceOffset=30; xml.typeElement=false; xml.typeWrapperElement=false

Table 5.84: DataConstr

Class	DataConstrRule			
Package	M2::AUTOSARTemplates::CommonStructure::GlobalConstraints			
Note	This meta-class represents the ability to express one specific data constraint rule.			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
constrLevel	Integer	0..1	attr	This attribute describes the category of a constraint. One of its functions is in the area of constraint violation, where it can be used from a certain level, to produce error messages. The lower the level, the more stringent the check. Used to distinguish hard or soft limits. Tags: xml.sequenceOffset=20
internalConstraints	InternalConstrs	0..1	aggr	Describes the limitations applicable on the internal domain (as opposed to the physical domain). Tags: xml.sequenceOffset=40
physConstraints	PhysConstrs	0..1	aggr	Describes the limitations applicable on the physical domain (as opposed to the internal domain). Tags: xml.sequenceOffset=30

Table 5.85: DataConstrRule

Class	PhysConstrs			
Package	M2::AUTOSARTemplates::CommonStructure::GlobalConstraints			
Note	This meta-class represents the ability to express physical constraints. Therefore it has (in opposite to InternalConstrs) a reference to a Unit.			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note

Attribute	Datatype	Mul.	Kind	Note
lowerLimit	Limit	0..1	ref	This specifies the lower limit of the constraint. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime xml.sequenceOffset=20
maxDiff	Numerical	0..1	attr	Maximum difference that is permitted between two consecutive values if the constraint is applied to an axis. Tags: xml.sequenceOffset=60
maxGradient	Numerical	0..1	attr	This element specifies the maximum slope that may be used in curves and maps. Tags: xml.sequenceOffset=50
monotony	MonotonyEnum	0..1	attr	This specifies the monotony constraints on the data object. Note that this applies only to curves and maps. Tags: xml.sequenceOffset=70
scaleConstraint (ordered)	ScaleConstr	*	aggr	This is one particular scale which contributes to the data constraints. Tags: xml.roleElement=true; xml.roleWrapperElement=true; xml.sequenceOffset=40; xml.typeElement=false; xml.typeWrapperElement=false
unit	Unit	0..1	ref	This is the unit to which the physical constraints relate to. In particular, it is the physical unit of the specified limits. Tags: xml.sequenceOffset=80
upperLimit	Limit	0..1	ref	This specifies the upper limit of the constraint. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime xml.sequenceOffset=30

Table 5.86: PhysConstrs

Class	InternalConstrs			
Package	M2::AUTOSARTemplates::CommonStructure::GlobalConstraints			
Note	This meta-class represents the ability to express internal constraints.			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
lowerLimit	Limit	0..1	ref	This specifies the lower limit of the constraint. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime xml.sequenceOffset=20

Attribute	Datatype	Mul.	Kind	Note
maxDiff	Numerical	0..1	attr	Maximum difference that is permitted between two consecutive values if the constraint is applied to an axis. Tags: xml.sequenceOffset=60
maxGradient	Numerical	0..1	attr	This element specifies the maximum slope that may be used in maps and curves. Tags: xml.sequenceOffset=50
monotony	MonotonyEnum	0..1	attr	This element specifies the monotony characteristics of the current internal or physical limits. The following table shows the monotony characteristics which are to be filled through the corresponding values. If the element has no contents or if it is omitted, "noMonotony" is the default content. Tags: xml.sequenceOffset=70
scaleConstr (ordered)	ScaleConstr	*	aggr	This is one particular scale which contributes to the data constraints. Tags: xml.roleElement=true; xml.roleWrapperElement=true; xml.sequenceOffset=40; xml.typeElement=false; xml.typeWrapperElement=false
upperLimit	Limit	0..1	ref	This specifies the upper limit defined by the constraint. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime xml.sequenceOffset=30

Table 5.87: InternalConstrs

Class	ScaleConstr			
Package	M2::AUTOSARTemplates::CommonStructure::GlobalConstraints			
Note	This meta-class represents the ability to specify constraints as a list of intervals (called scales).			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
desc	MultiLanguageOverviewParagraph	0..1	aggr	<desc> represents a general but brief description of the object in question. Tags: xml.sequenceOffset=30
lowerLimit	Limit	0..1	ref	This specifies the lower limit of the scale. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime xml.sequenceOffset=40

Attribute	Datatype	Mul.	Kind	Note
shortLabel	Identifier	0..1	ref	This element specifies a short name for the scaleConstr. This can for example be used to create more specific messages of a constraint checker. The constraints cannot be associated in the meta-model, therefore shortLabel is somehow a substitute for shortName. Tags: xml.sequenceOffset=20
upperLimit	Limit	0..1	ref	This specifies the upper limit of a the scale. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime xml.sequenceOffset=50
validity	ScaleConstrValidityEnum	0..1	attr	Specifies if the values defined by the scales are considered to be valid. If the attribute is missing then the default value is "VALID". Tags: xml.attribute=true

Table 5.88: ScaleConstr

Enumeration	ScaleConstrValidityEnum
Package	M2::AUTOSARTemplates::CommonStructure::GlobalConstraints
Note	This enumerator specifies the possible values of a scale.
Literal	Description
notAvailable	Currently invalid area The value usually is presented by the ECU but can currently not be performed due to e.g. initialization or temporary problems. Please note, that this behavior appears during runtime and cannot be handled while data is edited.
notDefined	Indicates an area which is marked in a specification (e.g. as reserved) Shall usually not be set by the ECU but is used by a tester to verify correct ECU.
notValid	The ECU cannot process the requested data.
valid	Current value is within a valid range and can be presented to user as is.

Table 5.89: ScaleConstrValidityEnum

Primitive	Limit			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::Primitive Types			
Note	This class represents the ability to express a numerical limit. Note that this is in fact a NumericalVariationPoint but has the additional attribute intervalType. Tags: xml.xsd.customType=LIMIT-VALUE; xml.xsd.pattern=(0x[0-9a-f+]) (0[0-7+]) (0b[0-1+]) ((([+-]?[1-9][0-9]+(\.[0-9]+)?[+-]?[0-9](\.[0-9]+)?)(E([+-]?[0-9]+)?))\.[0-9]+ NF -INF NaN; xml.xsd.type=string			
Attribute	Datatype	Mul.	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

Attribute	Datatype	Mul.	Kind	Note

Table 5.90: Limit

Enumeration	MonotonyEnum
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::Primitive Types
Note	This enumerator denotes the values for specification of monotony for e.g. curves.
Literal	Description
decreasing	This indicates that the related curve needs to be monotony decreasing.
increasing	This indicates that the related curve needs to be monotony increasing.
monotonous	This indicates that the values shall be monotonously decreasing or increasing, depending on the trend set by the first values of the series.
noMonotony	This indicates that the related curve needs not to be monotony.
strict Monotonous	This indicates that the values shall be strict monotonously decreasing or increasing, depending on the trend set by the first values of the series.
strictlyDe-creasing	This indicates that the related curve needs to be strictly monotony decreasing.
strictlyIn-creasing	This indicates that the related curve needs to be strictly monotony increasing.

Table 5.91: MonotonyEnum

[TPS_SWCT_01288] Interpretation of **PhysConstrs** and **InternalConstrs** by tools [**DataConstr** is an **ARElement** which can be reused by several data type specifications. Especially an **ImplementationDataType** and an **Application-DataType** which are mapped to each other, can refer to the same constraints or they can define their own constraints.

To avoid conflicts, in both cases **PhysConstrs** shall be interpreted by tools only with respect to application data types while **InternalConstrs** shall be interpreted only with respect to implementation data types. If either a physical or internal constraint is missing an existing **CompuMethod** can be used to calculate the missing information.]()

[TPS_SWCT_01289] Semantics of **Limit** [Technically, a **Limit** specifies a boundary of the interval of valid values for a given context (i.e. a data type). Please note that the boundary might or might not be part of the interval itself, i.e. the interval might be open or closed. From the formal point of view, the range represents all real numbers defined by:

$$\begin{aligned}
 \text{range} = & \{x \in \mathbb{R} \parallel \text{lowerLimit.value} < x < \text{upperLimit.value}\} \\
 & \cup \{\text{lowerLimit.value} \parallel \text{lowerLimit.intervalType} == \text{"CLOSED"}\} \\
 & \cup \{\text{upperLimit.value} \parallel \text{upperLimit.intervalType} == \text{"CLOSED"}\}
 \end{aligned}$$

]()

Please note that `Limit` inherits from `AbstractNumericalVariationPoint`. This means it is a number which may be subject to variability. For this reason, it is not possible to constrain the content already in the xml schema.

[constr_1191] Value of `Limit` shall yield a numerical value [After all variability is bound, the content obtained from a limit shall yield a numerical value.]()

Nevertheless it is not possible to distinguish on this level between float and integer values. Consequently [constr_1191] will not take the burden from an AUTOSAR tool to decide whether or not the value provided as a limit actually makes sense in any of the given contexts.

Enumeration	IntervalTypeEnum
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::Primitive Types
Note	This enumerator specifies the type of an interval.
Literal	Description
closed	The area is limited by the value given. The value itself is included.
open	The area is limited by the value given. The value itself is not included.

Table 5.92: IntervalTypeEnum

5.5.4 Addressing Methods

In an ECU there might be various methods to access a particular object (e.g measurement or calibration parameter) according to a given address. This variety might come from different kind of memory (near, far, ...) but also from indirections which are introduced by the compiler.

[TPS_SWCT_01290] `SwAddrMethod` [In order to allow a measurement and calibration system to access such objects `SwAddrMethods` are specified. Another purpose of this feature is to support the definition of abstract memory sections, i.e. to specify which variables shall be put together in the same sections in case of generated code (especially for data allocated by the RTE).

`SwAddrMethod` will be used to group data, for example, to cover the fact that sometimes it is required that one or more calibration parameters out of the overall collection of calibration parameters of a `SwComponentPrototype` respectively an AUTOSAR software component shall be placed in another memory location than the other parameters of the `SwComponentPrototype` respectively the AUTOSAR software component.]()

[TPS_SWCT_01291] Association of `MemorySection` with `SwAddrMethod` [In `Implementation` the particular `MemorySection` is associated with the `SwAddrMethod`. This association indicates that all objects of the associated addressing method shall be placed in the given memory section.]()

[TPS_SWCT_01456] Predefined values for `MemorySection.option` and `SwAddrMethod.option` [The following values of `MemorySection.option` and `SwAddrMethod.option` are predefined by AUTOSAR:

resetSafe This corresponds to variables of ECU-functions which values shall endure a ECU reset.

protected This corresponds to variables, constants, and code which shall not be accessible and modifiable from the outside without a security mechanism.

offline This corresponds to calibration parameters which shall not be modifiable during ECU operation.

coreGlobal This corresponds to variables, constants, and code which have to be accessible by any core in case of multi-core ECUs.

coreLocal This corresponds to variables, constants, and code which have to be accessible by one core in case of multi-core ECUs.

nvData This corresponds to variables of ECU-functions which shall be stored in non-volatile data. This option is applicable for memory used as a RAM Block managed by the NvM.

safetyQM This corresponds to variables, constants, and code without any safety integrity level and therefore having a QM rating.

safetyAsilA This corresponds to variables, constants, and code with the safety integrity level A.

safetyAsilB This corresponds to variables, constants, and code with the safety integrity level B.

safetyAsilC This corresponds to variables, constants, and code with the safety integrity level C.

safetyAsilD This corresponds to variables, constants, and code with the safety integrity level D.

]()

Obviously, the multiplicity of both the attribute `MemorySection.option` and `SwAddrMethod.option` allows for the appearance of more than one value. For example, a combination of the values `resetSafe`, `protected`, and `safetyAsilC` makes perfect sense on a particular list and can be used to express a meaning that combines the semantics of both values with each other.

However, this combination of values is not arbitrarily possible. It is therefore necessary to formulate a constraint that regulates the appearance of the safety-related values mentioned in [TPS_SWCT_01456].

In other words, it would not make any sense to attribute a given memory object with two different ASIL [26] values appearing on the same list.

If these values were combined on a particular list, the intended semantics would be ambiguous and could not clearly be determined. Therefore, [\[constr_1311\]](#) applies.

[constr_1311] Appearance of safety-related possible values of [MemorySection.option](#) or [SwAddrMethod.option](#) according to [\[TPS_SWCT_01456\]](#) [Any given list of values stored in the attributes [MemorySection.option](#) or [SwAddrMethod.option](#) shall at most include a single value out of the following list:

- **safetyQM**
- **safetyAsilA**
- **safetyAsilB**
- **safetyAsilC**
- **safetyAsilD**

]()

[constr_1381] Appearance of core-related possible values of [MemorySection.option](#) or [SwAddrMethod.option](#) according to [\[TPS_SWCT_01456\]](#) [Any given list of values stored in the attributes [MemorySection.option](#) or [SwAddrMethod.option](#) shall at most include a single value out of the following list:

- **coreGlobal**
- **coreLocal**

]()

[TPS_SWCT_01294] Missing [SwDataDefProps.swAddrMethod](#) [If the association [SwDataDefProps.swAddrMethod](#) is missing the object can be placed anywhere without restriction, e.g. using a default behavior of the RTE generator. Contradicting specifications (e.g. two different component types request different associations for one particular [SwAddrMethod](#)) shall be flagged as an error.]()

[TPS_SWCT_01292] Usage of [SwAddrMethod](#) in the context of a [DataPrototype](#) [Figure [5.43](#) illustrates the usage of [SwAddrMethod](#) in the context of a [DataPrototype](#). Note that the software component which defines the [DataPrototype](#) will in general not be the same to which the [Implementation](#) that actually contains the description of the [MemorySection](#) belongs.

The reason for this is that the resources for data allocated by the RTE will be described in the [Implementation](#) of the RTE. The indirection via [SwAddrMethod](#) makes this possible.]()

[TPS_SWCT_01293] RTE Generator has to derive the Memory Allocation Keyword [Please note that the RTE Generator has to derive the Memory Allocation Keyword used for [RunnableEntity](#)s and [BswSchedulableEntity](#)s from the [short-Name](#) of the [SwAddrMethod](#) only because the alignment defined in [MemorySection](#) is not known at contract phase.]()

[constr_2034] **SwAddrMethod** referenced by **RunnableEntity**s or **BswSchedulableEntity**s [**RunnableEntity**s and **BswSchedulableEntity**s shall not reference a **SwAddrMethod** which attribute **memoryAllocationKeywordPolicy** is set to **addrMethodShortNameAndAlignment**.]()

[constr_1402] **Applicability of core-related possible values of MemorySection.option or SwAddrMethod.option related to SwAddrMethod.sectionInitializationPolicy** [If the attribute **SwAddrMethod.option** or **MemorySection.option** is set to **coreLocal** then the attribute **SwAddrMethod.sectionInitializationPolicy** of the same **SwAddrMethod** respectively the **MemorySection.swAddrMethod** shall be either set to **INIT** or **CLEARED**.]()

The purpose of [constr_1402] is a reduction of the complexity of memory layouts and reduce the amount of memory gaps due to allocation restrictions.

Class	SwAddrMethod			
Package	M2::AUTOSARTemplates::CommonStructure::AuxillaryObjects			
Note	Used to assign a common addressing method, e.g. common memory section, to data or code objects. These objects could actually live in different modules or components. Tags: atp.recommendedPackage=SwAddrMethods			
Base	ARElement,ARObject,AtpBlueprint,AtpBlueprintable,CollectableElement,Identifiable,MultilanguageReferrable,PackageableElement,Referrable			
Attribute	Datatype	Mul.	Kind	Note
memoryAllocationKeywordPolicy	MemoryAllocationKeywordPolicyType	0..1	attr	Enumeration to specify the name pattern of the Memory Allocation Keyword.
option	Identifier	*	ref	This attribute introduces the ability to specify further intended properties of the MemorySection in with the related objects shall be placed. These properties are handled as to be selected. The intended options are mentioned in the list. In the Memory Mapping configuration, this option list is used to determine an appropriate MemMapAddressingModeSet.
sectionInitializationPolicy	SectionInitializationPolicyType	0..1	attr	Specifies the expected initialization of the variables (inclusive those which are implementing VariableDataPrototypes). Therefore this is an implementation constraint for initialization code of BSW modules (especially RTE) as well as the start-up code which initializes the memory segment to which the AutosarDataPrototypes referring to the SwAddrMethod's are later on mapped. If the attribute is not defined it has the identical semantic as the attribute value "INIT"
sectionType	MemorySectionType	0..1	attr	Defines the type of memory sections which can be associated with this addressing method.

Table 5.93: SwAddrMethod

Primitive	SectionInitializationPolicyType
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::PrimitiveTypes
Note	SectionInitializationPolicyType describes the intended initialization of MemorySections. The following values are standardized in AUTOSAR Methodology: • NO-INIT: No initialization and no clearing is performed. Such data elements shall not be read before one has written a value into it. • INIT: To be used for data that are initialized by every reset to the specified value (initValue). • POWER-ON-INIT: To be used for data that are initialized by "Power On" to the specified value (initValue). Note: there might be several resets between power on resets. • CLEARED: To be used for data that are initialized by every reset to zero. • POWER-ON-CLEARED: To be used for data that are initialized by "Power On" to zero. Note: there might be several resets between power on resets. Please note that the values are defined similar to the representation of enumeration types in the XML schema to ensure backward compatibility. Tags: xml.xsd.customType=SECTION-INITIALIZATION-POLICY-TYPE; xml.xsd.type=NMTOKEN

Table 5.94: SectionInitializationPolicyType

Enumeration	MemorySectionType
Package	M2::AUTOSARTemplates::CommonStructure::AuxillaryObjects
Note	Enumeration to specify the essential nature of the data which can be allocated in a common memory class by the means of the AUTOSAR Memory Mapping.
Literal	Description
calibrationVariables	This memory section is reserved for "virtual variables" that are computed by an MCD system during a measurement session but do not exist in the ECU memory.
calprm	To be used for calibratable constants of ECU-functions.
code	To be used for mapping code to application block, boot block, external flash etc.
configData	Constants with attributes that show that they reside in one segment for module configuration.
const	To be used for global or static constants.
excludeFromFlash	This memory section is reserved for "virtual parameters" that are taken for computing the values of so-called dependent parameter of an MCD system. Dependent Parameters that are not at the same time "virtual parameters" are allocated in the ECU memory. Virtual parameters, on the other hand, are not allocated in the ECU memory. Virtual parameters exist in the ECU Hex file for the purpose of being considered (for computing the values of dependent parameters) during an offline-calibration session.
var	To be used for global or static variables. The expected initialization is specified with the attribute sectionInitializationPolicy.

Table 5.95: MemorySectionType

Enumeration	MemoryAllocationKeywordPolicyType
Package	M2::AUTOSARTemplates::CommonStructure::AuxillaryObjects
Note	Enumeration to specify the name pattern of the Memory Allocation Keyword.
Literal	Description
addrMethod ShortName	The MemorySection shortNames of referring MemorySections and therefore the belonging Memory Allocation Keywords in the code are build with the shortName of the SwAddrMethod. This is the default value if the attribute does not exist.
addrMethod ShortName AndAlign-ment	The MemorySection shortNames of referring MemorySections and therefore the belonging Memory Allocation Keywords in the code are build with the shortName of the SwAddrMethod and a variable alignment postfix. Thereby the alignment postfix needs to be consistent with the alignment attribute of the related MemorySection.

Table 5.96: MemoryAllocationKeywordPolicyType

Primitive	AlignmentType
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::Primitive Types
Note	This primitive represents the alignment of objects within a memory section. The value is in number of bits or UNKNOWN (deprecated), 8 , 16, 32 UNSPECIFIED or BOOLEAN. Typical values for numbers are 8, 16, 32. Tags: xml.xsd.customType=ALIGNMENT-TYPE; xml.xsd.pattern=[1-9][0-9]* 0x[0-9a-f]* 0[0-7]* 0b[0-1]* UNSPECIFIED UNKNOWN B OOLEAN; xml.xsd.type=string

Table 5.97: AlignmentType

For more information on the specification of the `MemorySection` refer to [7].

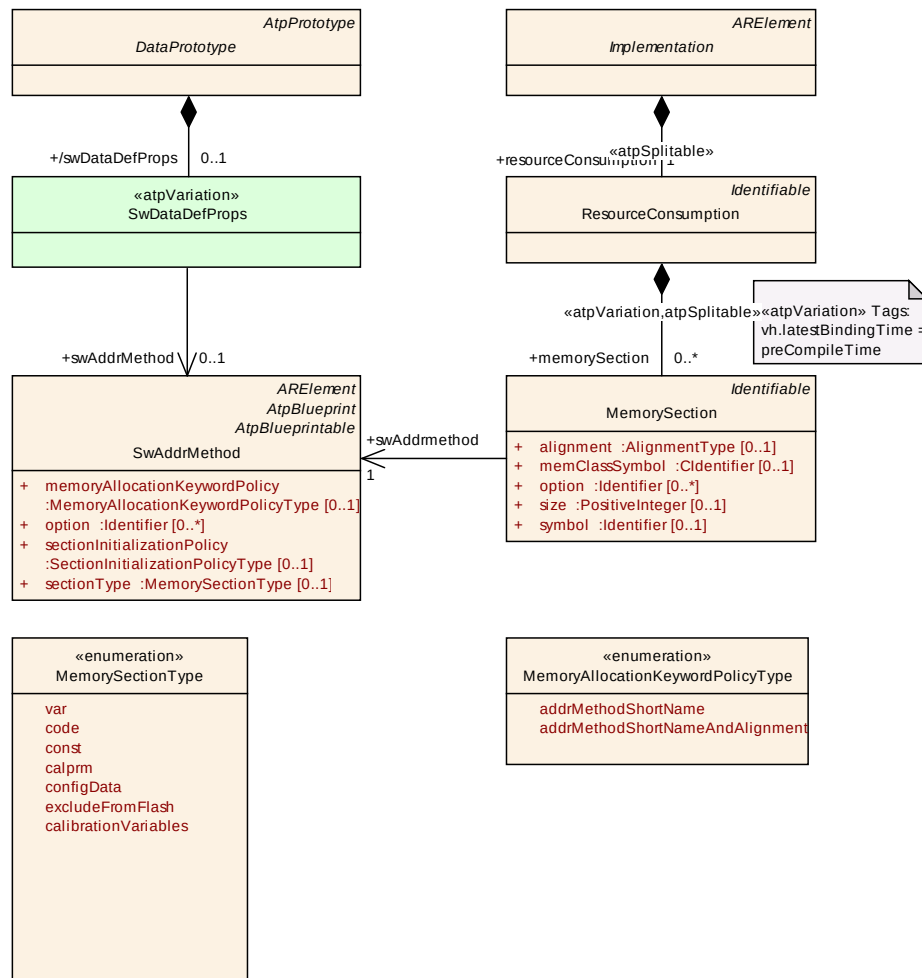


Figure 5.43: Assigning an address method to a memory section

5.5.5 Record Layouts

[TPS_SWCT_01295] **SwRecordLayout** [The **SwRecordLayout** describes how data is serialized in the memory of an ECU. This information is important with respect to the following aspects:

- to inform a measurement and calibration system how the data is serialized in the memory of an ECU
- to make sure that the software development results in the intended data structures
- to identify the proper interpolation routines

Via the **SwDataDefProps** a record-layout can be associated to a data entity. If the very same serialization approach is used for multiple **ApplicationDataTypes** all of these may refer to the same **SwRecordLayout** even if the size of the data is different.

5.5.5.1 Specifying Record Layouts

As mentioned above, the purpose of record layout is to specify how an object (e.g. a calibration parameter) is serialized in memory of an ECU. The canonical approach for this is to define nested groups ([SwRecordLayoutGroup](#)).

These groups indicate the structure of the corresponding [Implementation-DataType](#). The serialization is then executed by iterating over the axes of a curve, a map, or iterating along a string. The contents of such a record layout group ([SwRecordLayoutGroupContent](#)) is a mixture of (thus nested) groups and values ([SwRecordLayoutV](#)).

These values refer to particular properties of the object (e.g. value, count, ...). By application of this pattern, the serialization of any complex object can be specified.

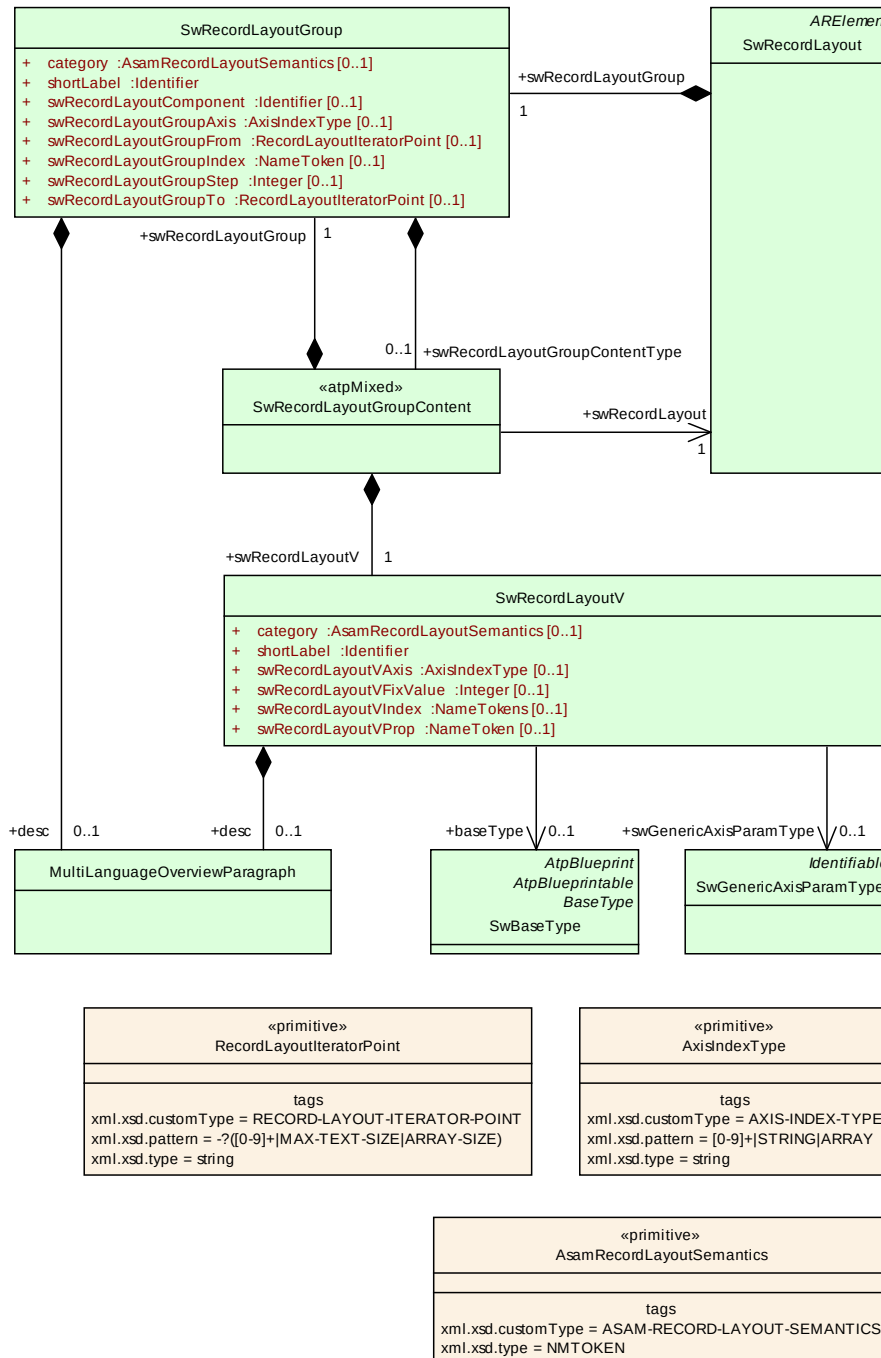


Figure 5.44: Specification of a record layout

Class	SwRecordLayout			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::RecordLayout			
Note	Defines how the data objects (variables, calibration parameters etc.) are to be stored in the ECU memory. As an example, this definition specifies the sequence of axis points in the ECU memory. Iterations through axis values are stored within the sub-elements swRecordLayoutGroup. Tags: atp.recommendedPackage=SwRecordLayouts			
Base	ARElement , ARObject , CollectableElement , Identifiable , Multilanguage Referrable , PackageableElement , Referrable			
Attribute	Datatype	Mul.	Kind	Note
swRecordLayoutGroup	SwRecordLayoutGroup	1	aggr	This is the top level record layout group. Tags: xml.roleElement=true; xml.roleWrapperElement=false; xml.sequenceOffset=20; xml.typeElement=false; xml.typeWrapperElement=false

Table 5.98: SwRecordLayout

Class	SwRecordLayoutV			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::RecordLayout			
Note	This element specifies which values are stored for the current SwRecordLayoutGroup. If no baseType is present, the SwBaseType referenced initially in the parent SwRecordLayoutGroup is valid. The specification of swRecordLayoutVAxis gives the axis of the values which shall be stored in accordance with the current record layout SwRecordLayoutGroup. In swRecordLayoutVProp one can specify the information which shall be stored.			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
desc	MultiLanguageOverviewParagraph	0..1	aggr	This aggregation allows for a brief description about the particular record layout value which can help to identify the entry. In-depth documentation should be added to the introduction of the surrounding record layout. Tags: xml.sequenceOffset=20
category	AsamRecordLayoutSemantics	0..1	attr	This attribute denotes the semantics in particular in terms of the corresponding A2L-Keyword. This is to support the mapping of the more general record layouts in AUTOSAR/MSR to the specific A2L keywords. It is possible to express the specific semantics of A2L RecordLayout keywords in swRecordLayoutGroup but not always vice versa. Therefore the mapping is provided in this optional attribute. Tags: xml.sequenceOffset=5
baseType	SwBaseType	0..1	ref	This association allows to refer to a base type in case a specific encoding is intended. If no base type is referred, the base type referenced initially in the corresponding DataPrototype is to be used. Tags: xml.sequenceOffset=30

Attribute	Datatype	Mul.	Kind	Note
shortLabel	Identifier	1	ref	This attribute specifies a name which can be used e.g. when ECU code is generated from the record layout value. Tags: xml.sequenceOffset=3
swGenericAxisParamType	SwGenericAxisParamType	0..1	ref	This association supports the case that a value from a generic axis definition shall be stored. This value is denoted by a particular generic axis parameter type. Tags: xml.sequenceOffset=70
swRecordLayoutVAxis	AxisIndexType	0..1	attr	This attribute gives the index of the axis of which values that are stored in the record. swRecordVIndex refers to the symbolic names of the iterators for which the axis value shall be stored in the record. In case of nested iterators (mainly for multidimensional objects) the iterator names are specified as whitespace-separated names. These symbolic names relate to swRecordLayoutGroupIndex. The iterators are processed from left to right in such a manner that they symbolize the loop index from the outside to the inside. It is considered an error if more components are specified than axes exist in the related ApplicationDataType. Tags: xml.sequenceOffset=40
swRecordLayoutVFixValue	Integer	0..1	attr	This attribute specifies the filler character for the current record layout, in the form of hex digits. It is also used to specify the fix value for e.g. FIXRIGHTDIFF. Tags: xml.sequenceOffset=80
swRecordLayoutVIndex	NameTokens	0..1	attr	The symbolic value for iteration, or the symbolic values separated by whitespaces, refer to the symbolic values given in swRecordLayoutGroupIndex . The iterators are processed from left to right, in such a manner that they symbolize the loop index from the outside to the inside. It is considered an error if the record layout is referenced by an entity which has less number of axes than index names referenced here. Tags: xml.sequenceOffset=60

Attribute	Datatype	Mul.	Kind	Note
swRecordLayoutVProp	NameToken	0..1	attr	This attribute describes the kind of values to be stored. More details see below. The standardized values foreseen for this attribute are defined in [TPS_SWCT_01489]. Tags: xml.sequenceOffset=50

Table 5.99: SwRecordLayoutV

Class	SwRecordLayoutGroup			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::RecordLayout			
Note	Specifies how a record layout is set up. Using SwRecordLayoutGroup it recursively models iterations through axis values. The subelement swRecordLayoutGroupContentType may reference other SwRecordLayouts, SwRecordLayoutVs and SwRecordLayoutGroups for the modeled record layout.			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
desc	MultiLanguageOverviewParagraph	0..1	aggr	This aggregation allows a brief description about the particular record layout group which can help to identify the entry. In-depth documentation should be added to the introduction of the surrounding record layout. Tags: xml.sequenceOffset=20
category	AsamRecordLayoutSemantics	0..1	attr	This attribute denotes the semantics in particular in terms of the corresponding A2L-Keyword. This is to support the mapping of the more general record layouts in AUTOSAR/MSR to the specific A2I keywords. It is possible to express the specific semantics of A2I recordlayout keywords in swRecordlayoutGroup but not always vice versa. Therefore the mapping is provided in this optional attribute. Tags: xml.sequenceOffset=5
shortLabel	Identifier	1	ref	This attribute specifies a name which can be used e.g. when ECU code is generated from the record layout group. Tags: xml.sequenceOffset=3

Attribute	Datatype	Mul.	Kind	Note
swGenericAxisParamType	SwGenericAxisParamType	0..1	ref	This association allows to specify record layout groups to iterate over generic axis parameters. For example, if the generic axis parameter is an array, the record layout group will iterate over this array. Obviously, the axis referred to by swRecordLayoutGroupAxis shall be a generic axis in which the referenced SwGenericAxisType is aggregated. Tags: xml.sequenceOffset=50
swRecordLayoutComponent	Identifier	0..1	ref	This attribute is used to denote the component to which the group in question applies. Thus, the record layout supports structured objects. This secures independence from the sequence of components, because they can be referred to via name. Tags: xml.sequenceOffset=90
swRecordLayoutGroupAxis	AxisIndexType	0..1	attr	This attribute specifies the iteration axis number for a SwRecordLayoutGroup. The current record layout group then refers exactly to the axis with this number. This means that the values are taken by iterating along the thus referenced axis. Tags: xml.sequenceOffset=30
swRecordLayoutGroupContentType	SwRecordLayoutGroupContent	0..1	aggr	This is the contents of the recordLayout which is produced for every step of iteration. Tags: xml.roleElement=false; xml.roleWrapperElement=false; xml.sequenceOffset=100; xml.typeElement=false; xml.typeWrapperElement=false
swRecordLayoutGroupFrom	RecordLayoutIteratorPoint	0..1	attr	This attribute specifies the iterator index for the point in the axis from which a record layout group is commenced. Negative values are also possible, i.e. the value -4 counts from the fourth value from the end. If this property is missing, the iteration starts with '1'. Tags: xml.sequenceOffset=60
swRecordLayoutGroupIndex	NameToken	0..1	attr	This attribute attributes a symbolic name to the iterator of the superimposed record layout group. This can be referenced as a loop index in contained SwRecordLayoutV elements. Tags: xml.sequenceOffset=40

Attribute	Datatype	Mul.	Kind	Note
swRecordLayoutGroupStep	Integer	0..1	attr	This attribute specifies the step width for the iterator index that is used for the current record layout group. Note that negative values are also possible, in case of the starting point is higher than the endpoint. If the property is missing, the step width is "1". Tags: xml.sequenceOffset=80
swRecordLayoutGroupTo	RecordLayoutIteratorPoint	0..1	attr	This attribute specifies the end point for the iteration. Negative values are also possible, i.e. the value -4 counts up to the fourth value from the end. If this property is not there, the iteration ends at "-1" which is the last element. Note that depending on the arraySizeSemantics of SwTextProps the iteration ends at the value specified in swMaxTextSize. Tags: xml.sequenceOffset=70

Table 5.100: SwRecordLayoutGroup

Class	«atpMixed» SwRecordLayoutGroupContent			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::RecordLayout			
Note	This is the contents of a RecordLayout which is inserted for every iteration. Note that since this is atpMixed, multiple properties can be inserted for each iteration.			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
swRecordLayout	SwRecordLayout	1	ref	This association allows to support reusable "sub"-record layouts. In particular, the contents of the referenced record layout shall be used as if the record layout group in the referenced record layout was aggregated in the current record layout group. So, semantically it would be equivalent to replace the particular association with an aggregation of the swRecordLayoutGroup of the referenced SwRecordLayout. Tags: xml.sequenceOffset=110
swRecordLayoutGroup	SwRecordLayoutGroup	1	aggr	This aggregation provides support for nested iterations. For example, if a map is to be handled, then we might have two nested SwRecordLayoutGroups, one for the x-axis and one for the y-axis. The inner iteration runs faster. Tags: xml.sequenceOffset=130

Attribute	Datatype	Mul.	Kind	Note
swRecordLayoutV	SwRecordLayoutV	1	aggr	Particular Value specification for this record layout group. Tags: xml.sequenceOffset=120

Table 5.101: SwRecordLayoutGroupContent

[constr_1264] Iteration along output axis is only supported for **VALUE** and **VAL_BLK** [swRecordLayoutVIndex in SwRecordLayoutV cannot be 0 for any value of SwRecordLayoutV.category other than **VALUE** and **VAL_BLK**.]()

For **CURVE**, **MAP**, etc. the iteration shall be performed along the input axis.

Primitive	AxisIndexType
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::RecordLayout
Note	This meta-class specifies an axis in a curve/map data object. The index satisfies the following convention: • 0 output "axis" • 1 input axis 1 (X input axis e.g. of a CURVE) • 2 input axis 2 (Y input axis e.g. of a MAP) • 3 input axis 3 (Z input axis e.g. of a CUBOID) • 4 input axis 3 (Z4 input axis e.g. of a CUBE_4) • 5 input axis 3 (Z5 input axis e.g. of a CUBE_5) • 6..9 etc. The output "axis" provides access to the output value of the parameter. Note that this access is usually performed via an index according to the input axis. In addition to this, the Values STRING and ARRAY support specific iterations. Tags: xml.xsd.customType=AXIS-INDEX-TYPE; xml.xsd.pattern=[0-9]+ STRING ARRAY; xml.xsd.type=string

Table 5.102: AxisIndexType

Primitive	RecordLayoutIteratorPoint
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::RecordLayout
Note	This meta-class denotes a start / endpoint for the iteration of a SwRecordLayoutGroup. It can be an integer or one of the keywords MAX-TEXT-SIZE ARRAY-SIZE. Note that negative numbers are counted backwards. Therefore e.g. -1 refers to the last value. Tags: xml.xsd.customType=RECORD-LAYOUT-ITERATOR-POINT; xml.xsd.pattern=-?([0-9]+ MAX-TEXT-SIZE ARRAY-SIZE); xml.xsd.type=string

Table 5.103: RecordLayoutIteratorPoint

[TPS_SWCT_01489] Standardized values of [SwRecordLayoutV.swRecordLayoutVProp](#) [[SwRecordLayoutV.swRecordLayoutVProp](#) describes the type of values to be stored. The standardized values for [SwRecordLayoutV.swRecordLayoutVProp](#) are listed in Table 5.104.]()

Property	Description
VALUE	The value of the axis for the current iterator point. This is e.g. the particular point on an input-axis, but also the particular character in a string.
COUNT	The amount of values of the axis
LEFTDIFF	The difference to the previous axis point
RIGHTDIFF	The difference to the next axis point
DIST	The distance value of this axis in case of a fixed axis with distance specification
SHIFT	The shift value of this axis in case of a fixed axis with shift/offset
OFFSET	The offset value of this axis in case of a fixed axis with shift/offset
SOURCE-ADR	The address of the source of this axis (Note that this does not apply to the value axis)
RESULT-ADR	The address of the result for this axis (note that this does not apply to input axis)
ADDRESS	The address of the axis point
FILL	Fill with the hex value specified as contents of swRecordLayoutVFixValue
FIXLEFTDIFF	Difference between this and a fixed left-hand value specified in swRecordLayoutVFixValue
FIXRIGHTDIFF	Difference between this and a fixed right-hand value specified in swRecordLayoutVFixValue

Table 5.104: swRecordLayoutVProp

Figure 5.45 and Figure 5.46 illustrate most of these properties.

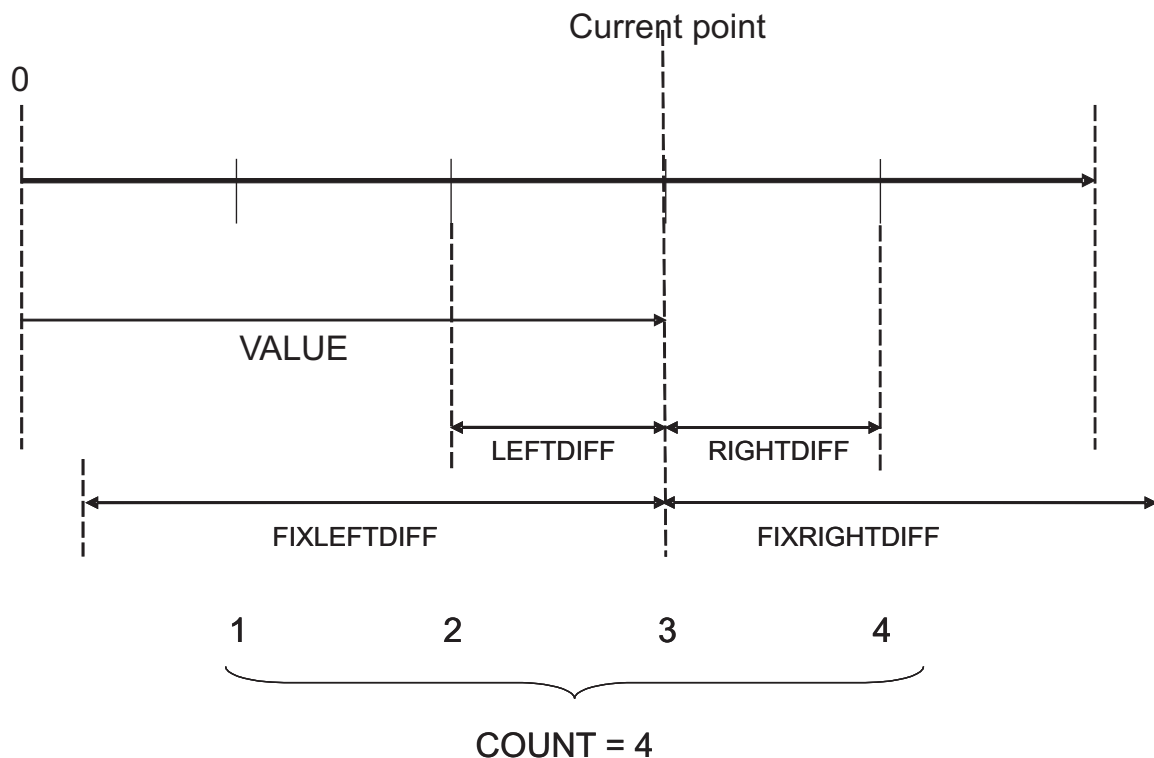


Figure 5.45: Values for swRecordLayoutVProp for individual axis

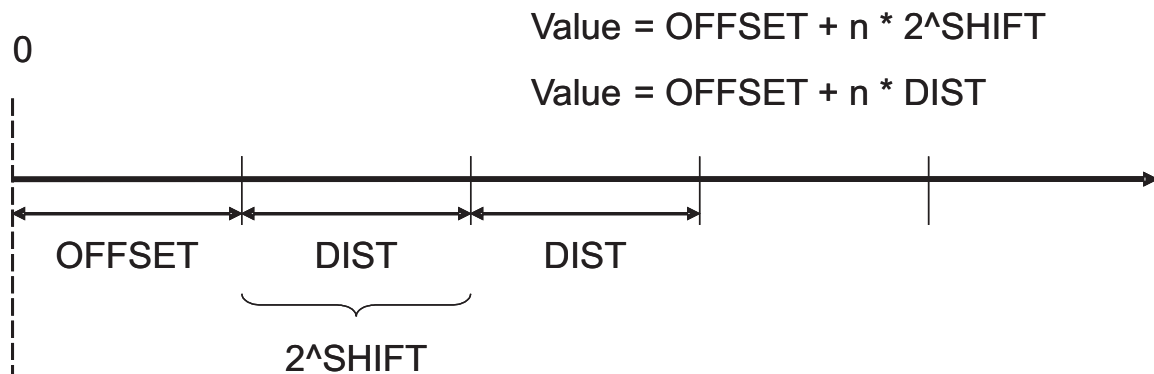


Figure 5.46: Values for swRecordLayoutVProp for fixed axis

[TPS_SWCT_01296] Different approaches of ASAM MCD-2MC and AUTOSAR with respect to [SwRecordLayout](#) [ASAM MCD-2D specification (also known as A2L, resp. ASAP) uses keywords in record layouts where MSR/AUTOSAR uses the more generic approach specified here.

It may happen that this generic approach cannot always be safely mapped to the A2L keywords. Therefore [SwRecordLayoutV.category](#) as well as [SwRecordLayoutGroup.category](#) can assist the conversion to the current A2L format. `]()`

Primitive	AsamRecordLayoutSemantics
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::RecordLayout
Note	This meta-class is used to denote the semantics in particular in terms of the corresponding A2L-Keyword. This is to support the mapping of the more general record layouts in AUTOSAR/MSR to the specific A2L keywords. It is possible to express the specific semantics of A2L RecordLayout keywords in SwRecordLayoutGroup but not always vice versa. Therefore the mapping is provided in this optional attribute. It is specified as NMTOKEN to reduce the direct dependency of ASAM an AUTOSAR standards. Tags: xml.xsd.customType=ASAM-RECORD-LAYOUT-SEMANTICS; xml.xsd.type=NMTOKEN

Table 5.105: AsamRecordLayoutSemantics

The values of [SwRecordLayoutV.category](#) resp. [SwRecordLayoutGroup.category](#) can, for example, be taken from the ASAM MCD 2D specification provided in [25]. Examples are such as INDEX_INCR, INDEX_DECR, COLUMN_DIR, ROW_DIR, ALTERNATE_WITH_X, ALTERNATE_WITH_Y, ALTERNATE_CURVES.

The consistency of these values of [SwRecordLayoutV.category](#) resp. [SwRecordLayoutGroup.category](#) with the structure of the [SwRecordLayout](#) shall be ensured by the author of the SwRecord Layout.

Note that there are keywords in A2L bound to a calibration parameter which in MSR/AUTOSAR are represented by the [SwRecordLayout](#) (DEPOSIT etc.).

The following XML fragment provides an example for a `SwRecordLayout` for a curve. Note that in this case recognizing the patterns represented by the A2L-Keywords (shown in XML-Comment) is pretty straight forward, even if the keywords were not provided in the `SwRecordLayoutV.category` as well as `SwRecordLayoutGroup.category`.

Listing 5.12: Example for RecordLayout of a curve

```
<SW-RECORD-LAYOUT>
  <SHORT-NAME>RecordLayoutCurve</SHORT-NAME>
  <SW-RECORD-LAYOUT-GROUP>
    <SW-RECORD-LAYOUT-V><!-- SRC_ADDR_X -->
      <SHORT-LABEL>srcAdr</SHORT-LABEL>
      <SW-RECORD-LAYOUT-V-PROP>SOURCE-ADR</SW-RECORD-LAYOUT-V-PROP>
    </SW-RECORD-LAYOUT-V>
    <SW-RECORD-LAYOUT-V><!-- NO_AXIS_PTS_X -->
      <SHORT-LABEL>noOfAxisPts</SHORT-LABEL>
      <SW-RECORD-LAYOUT-V-PROP>COUNT</SW-RECORD-LAYOUT-V-PROP>
      <SW-RECORD-LAYOUT-V-INDEX>1</SW-RECORD-LAYOUT-V-INDEX>
    </SW-RECORD-LAYOUT-V>
    <SW-RECORD-LAYOUT-GROUP><!-- AXIS_PTS_X -->
      <SHORT-LABEL>xPts</SHORT-LABEL>
      <CATEGORY>INDEX_INCR</CATEGORY>
      <SW-RECORD-LAYOUT-GROUP-AXIS>1</SW-RECORD-LAYOUT-GROUP-AXIS>
      <SW-RECORD-LAYOUT-GROUP-FROM>1</SW-RECORD-LAYOUT-GROUP-FROM>
      <SW-RECORD-LAYOUT-GROUP-TO>-1</SW-RECORD-LAYOUT-GROUP-TO>
      <SW-RECORD-LAYOUT-V>
        <SHORT-LABEL>xPt</SHORT-LABEL>
        <SW-RECORD-LAYOUT-V-AXIS>1</SW-RECORD-LAYOUT-V-AXIS> <!--
          AXIS_PTS_X -->
        <SW-RECORD-LAYOUT-V-PROP>VALUE</SW-RECORD-LAYOUT-V-PROP>
      </SW-RECORD-LAYOUT-V>
    </SW-RECORD-LAYOUT-GROUP>
    <SW-RECORD-LAYOUT-GROUP>
      <SHORT-LABEL>values</SHORT-LABEL><!-- FNC_VALUES -->
      <CATEGORY>COLUMN_DIR</CATEGORY>
      <SW-RECORD-LAYOUT-GROUP-AXIS>0</SW-RECORD-LAYOUT-GROUP-AXIS>
      <SW-RECORD-LAYOUT-GROUP-FROM>1</SW-RECORD-LAYOUT-GROUP-FROM>
      <SW-RECORD-LAYOUT-GROUP-TO>-1</SW-RECORD-LAYOUT-GROUP-TO>
      <SW-RECORD-LAYOUT-V>
        <SHORT-LABEL>value</SHORT-LABEL>
        <SW-RECORD-LAYOUT-V-AXIS>0</SW-RECORD-LAYOUT-V-AXIS><!--
          FNC_VALUES -->
        <SW-RECORD-LAYOUT-V-PROP>VALUE</SW-RECORD-LAYOUT-V-PROP>
      </SW-RECORD-LAYOUT-V>
    </SW-RECORD-LAYOUT-GROUP>
  </SW-RECORD-LAYOUT-GROUP>
</SW-RECORD-LAYOUT>
```

5.5.5.2 RecordLayouts and DataTypes

[constr_1027] Types for record layouts [Because `ParameterDataPrototypes` have a `«isOfType»`-relation to `ApplicationDataTypes` or `Implementation-`

`DataType`s the related data types shall properly match to the details as specified in `swDataDefProps`. `]()`

This is exemplified in figure 5.47.

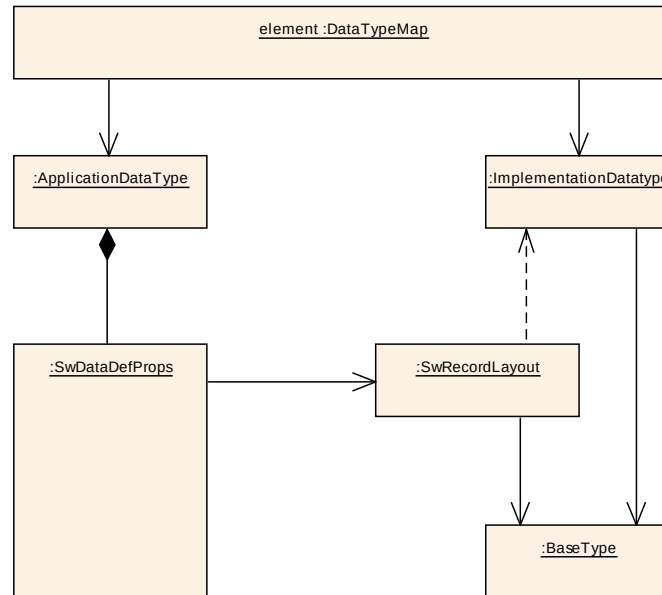


Figure 5.47: Dependency of `AutosarDataTypes` and `SwRecordLayouts`

[TPS_SWCT_01297] Compliance of `ApplicationDataTypes` or `ImplementationDataTypes` to `swDataDefProps` `[` In order to maintain this compliance the following options exist:

- Manually create `ImplementationDataTypes` from corresponding `ApplicationDataTypes` and the referenced `SwRecordLayouts`
- Automatically create `ImplementationDataTypes` according to the existing definition of `SwRecordLayouts`. This could be performed by a model transformation according to the algorithm shown below.

`]()`

[TPS_SWCT_01298] Computing `SwRecordLayout` from `ImplementationDataTypes` is not possible `[` Note that computing `SwRecordLayouts` from `ImplementationDataTypes` is not really possible because the particular semantics of the components is not available (`swRecordLayoutVProp`). `]()`

Figure 5.48 and figure 5.49 illustrate how data types can be derived from `SwRecordLayouts`.

The “blue” data types are derived from the record layout. These diagrams illustrate in particular the fact that on the level of `ApplicationDataType` even complex entities such as curves and maps appear as somehow primitive. The inner details of such entities are handled e.g. by service libraries.

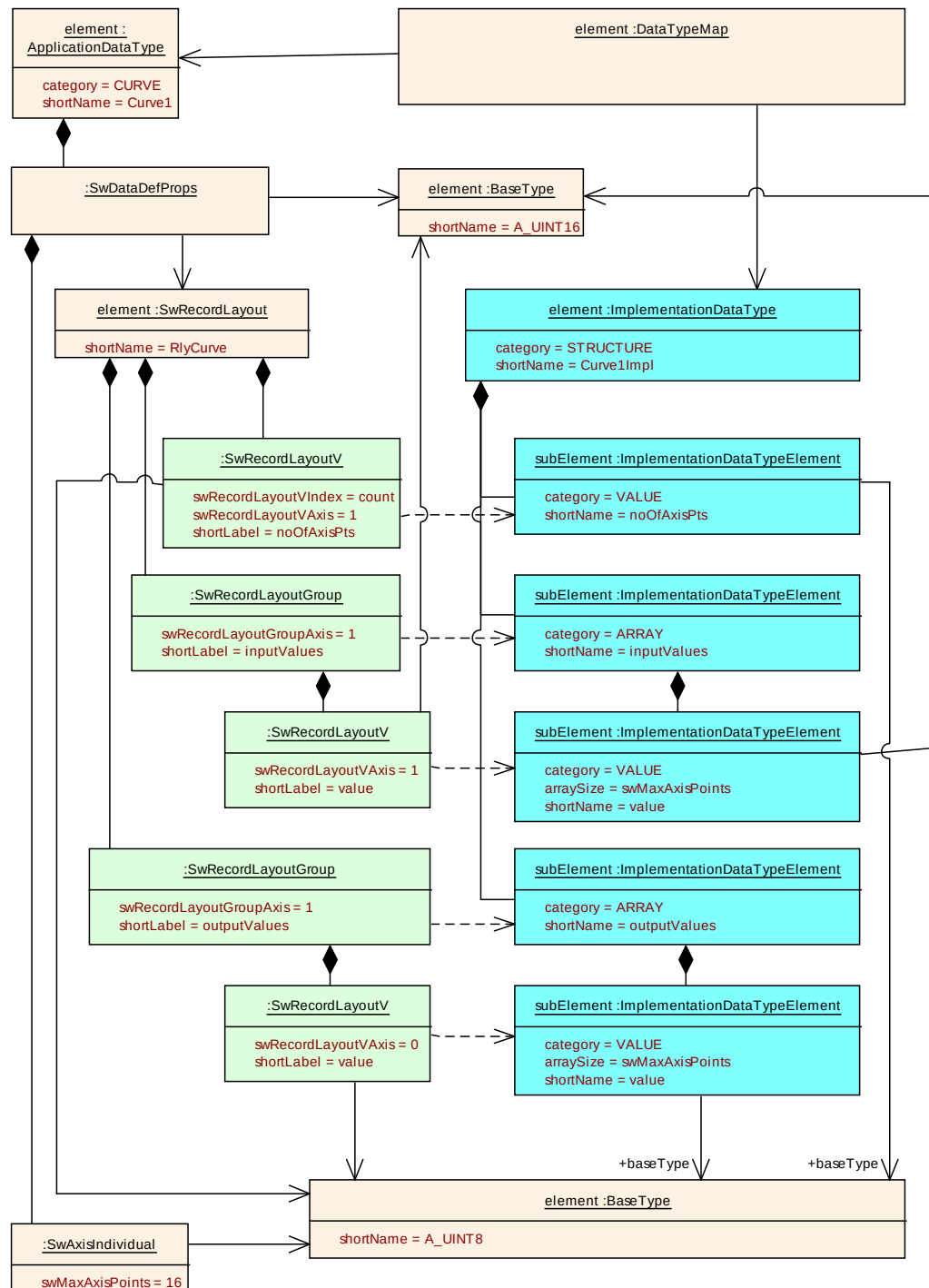


Figure 5.48: Curve implemented as two consecutive arrays

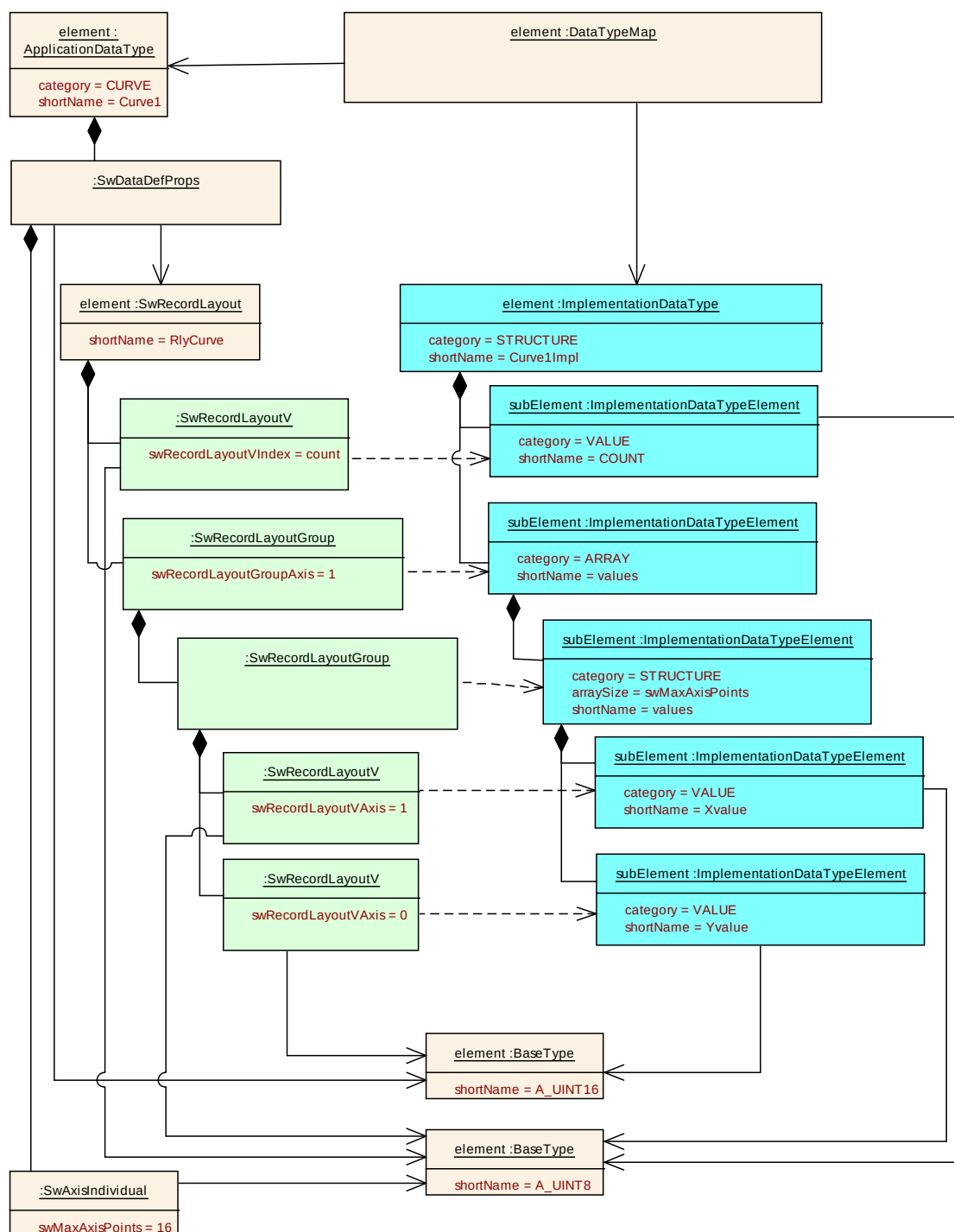


Figure 5.49: Curve implemented as array of value pairs

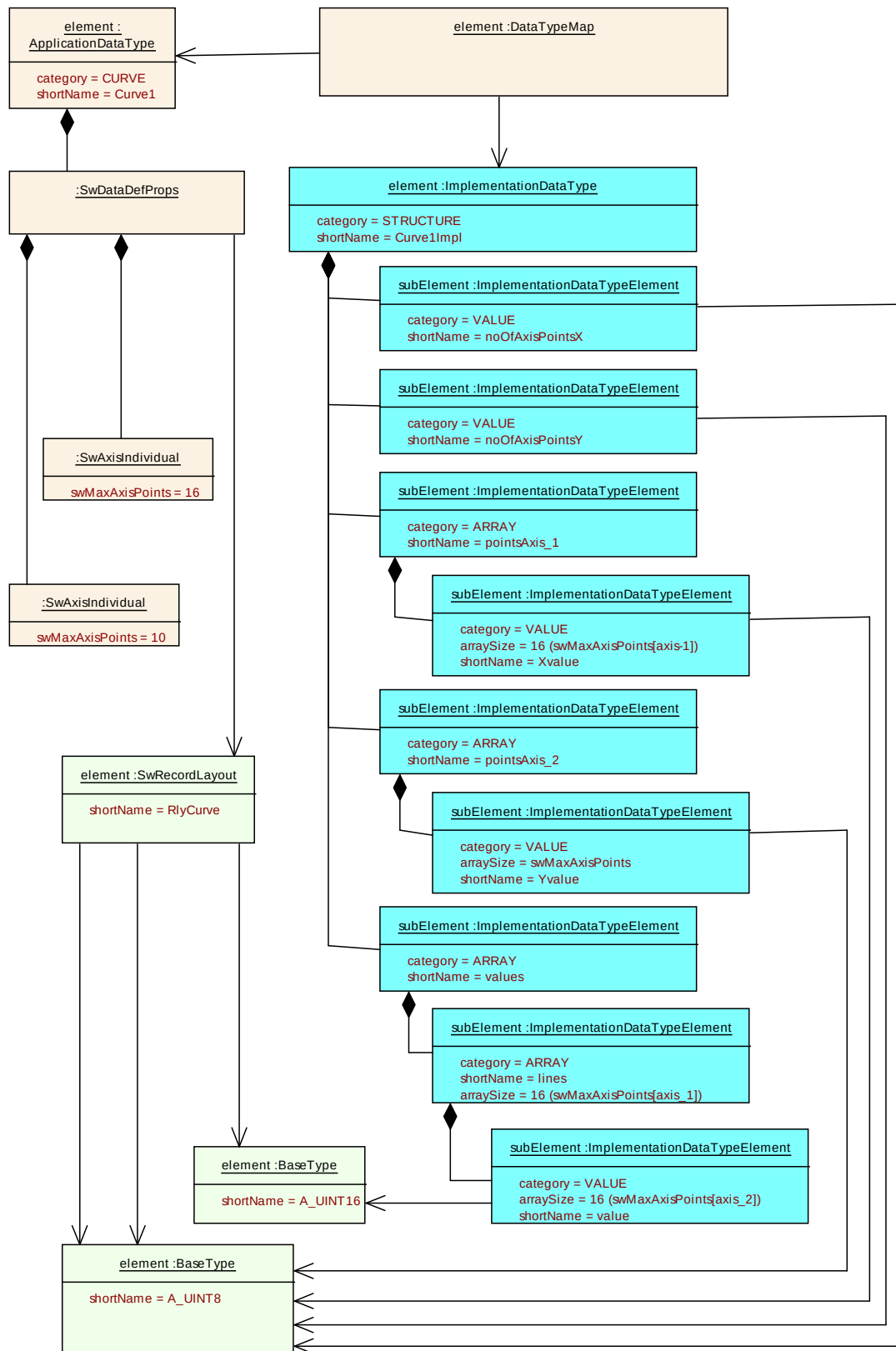


Figure 5.50: Record layout and data type for a map

The algorithm to generate the desired data types is illustrated in the following two diagrams.

We create an `ImplementationDataType` for each `ApplicationDataType`. Figure 5.51 illustrates how to map the details.

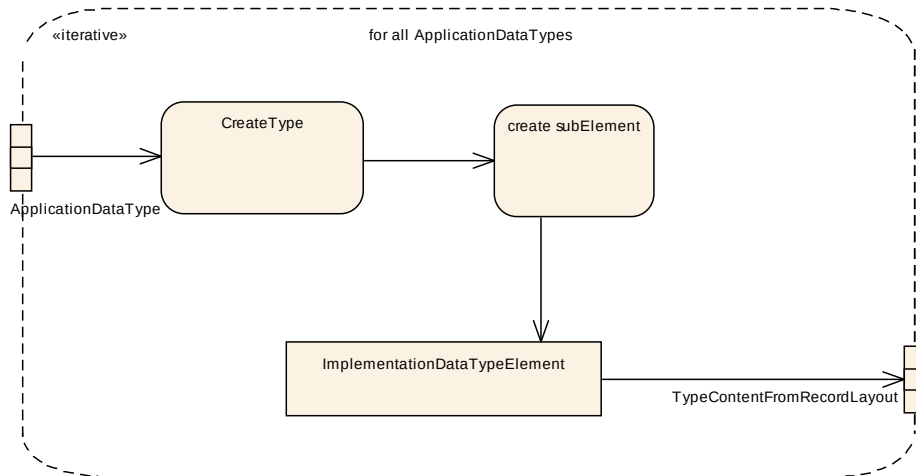


Figure 5.51: algorithm to map the details of an application data type to the corresponding implementation data type according to the record layout

[TPS_SWCT_01299] Relation of `swRecordLayoutGroup` to `subElement` [For each `swRecordLayoutGroup` an appropriate `subElement` shall be created.

This sub element is then refined according to the approach sketched in figure 5.52. The algorithm shall be recursively applied applied to the newly created `ImplementationDataTypeElements`. As the record layout groups are nested, this recursion yields the complete structure in the `ImplementationDataType`.]()

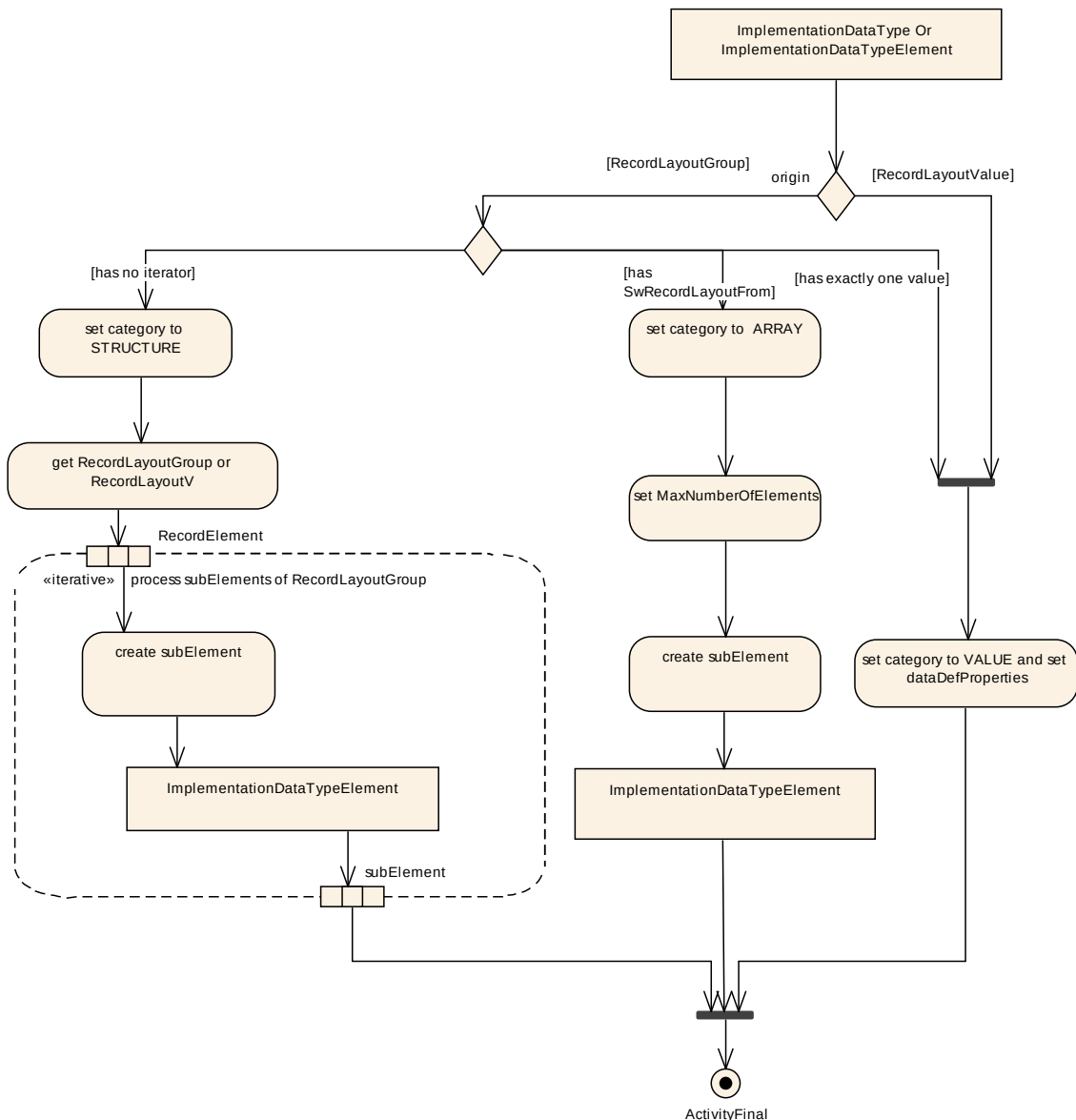


Figure 5.52: refining subElements

5.5.5.3 Record Layouts and Interpolation Routines

[TPS_SWCT_01300] Relationship between record layouts and interpolation routines The relationship between record layouts and interpolation routines can be specified in [InterpolationRoutineMappingSet](#).

The interpolation routine is represented as [BswModuleEntry](#) and implements a particular interpolation method which is denoted in the value of [InterpolationRoutine.shortLabel](#).

The intended interpolation method is denoted in the value of attribute [SwDataDefProps.swInterpolationMethod](#). `()`

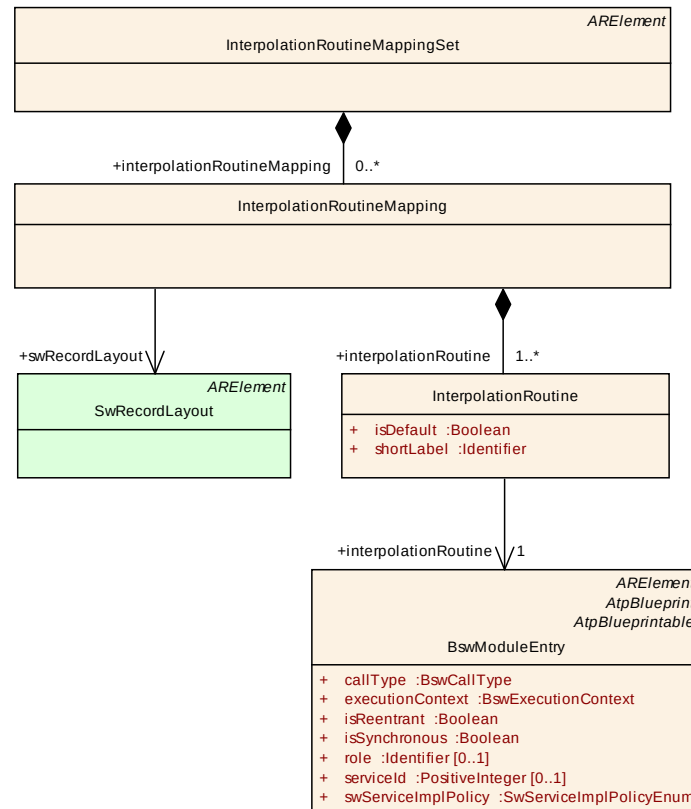


Figure 5.53: Mapping of Record Layouts and Interpolation Routines

Class	InterpolationRoutineMappingSet			
Package	M2::AUTOSARTemplates::SWComponentTemplate::MeasurementAndCalibration::InterpolationRoutineMappingSet			
Note	This meta-class specifies a set of interpolation routine mappings. Tags: atp.recommendedPackage=InterpolationRoutineMappingSets			
Base	ARElement, ARObjct, CollectableElement, Identifiable, Multilanguage Referrable, PackageableElement, Referrable			
Attribute	Datatype	Mul.	Kind	Note
interpolationRoutineMapping	InterpolationRoutineMapping	*	aggr	This specifies one particular mapping of recordlayout and its matching interpolationRoutines.

Table 5.106: InterpolationRoutineMappingSet

Class	InterpolationRoutineMapping			
Package	M2::AUTOSARTemplates::SWComponentTemplate::MeasurementAndCalibration::InterpolationRoutineMappingSet			
Note	This meta-class provides a mapping between one record layout and its matching interpolation routines. This allows to formally specify the semantics of the interpolation routines. The use case is such that the curves/Maps define an interpolation method. This mapping table specifies which interpolation routine implements methods for a particular record layout. Using this information, the implementer of a software-component can select the appropriate interpolation routine.			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
interpolationRoutine	InterpolationRoutine	1..*	aggr	This is one particular interpolation routine which is mapped to the record layout.
swRecordLayout	SwRecordLayout	1	ref	This refers to the record layout which is mapped to interpolation routines.

Table 5.107: InterpolationRoutineMapping

Class	InterpolationRoutine			
Package	M2::AUTOSARTemplates::SWComponentTemplate::MeasurementAndCalibration::InterpolationRoutineMappingSet			
Note	This represents an interpolation routine taken to evaluate the contents of a curve or map against a specific input value.			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
interpolationRoutine	BswModuleEntry	1	ref	This specifies a BswModuleEntry which implements the current interpolation method for the given record layout. Tags: xml.sequenceOffset=30
isDefault	Boolean	1	attr	This specifies if the current interpolationMethod is the default for the referenced record layout. Tags: xml.sequenceOffset=20
shortLabel	Identifier	1	ref	This is the name of the interpolation method which is implemented by the referenced bswModuleEntry. It corresponds to swInterpolationMethod in SwDataDefProps. Tags: xml.sequenceOffset=10

Table 5.108: InterpolationRoutine

5.6 Specification of Constant Values

5.6.1 Overview

[TPS_SWCT_01177] Assignment of constant values [Constant values can be assigned to a meta-class by aggregating the meta-class [ValueSpecification](#). This aggregation can be used in two ways:

1. by referencing to a reusable [ConstantSpecification](#) which contains another [ValueSpecification](#)
2. or through an inline aggregation of a value specification of various kind.

]([RS_SWCT_03175](#))

Class	ConstantSpecification			
Package	M2::AUTOSARTemplates::CommonStructure::Constants			
Note	Specification of a constant that can be part of a package, i.e. it can be defined stand-alone. Tags: atp.recommendedPackage=ConstantSpecifications			
Base	ARElement , ARObject , CollectableElement , Identifiable , Multilanguage , Referrable , PackageableElement , Referrable			
Attribute	Datatype	Mul.	Kind	Note
valueSpec	ValueSpecification	1	aggr	Specification of an expression leading to a value for this constant.

Table 5.109: ConstantSpecification

Class	ValueSpecification (abstract)			
Package	M2::AUTOSARTemplates::CommonStructure::Constants			
Note	Base class for expressions leading to a value which can be used to initialize a data object.			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
shortLabel	Identifier	0..1	ref	This can be used to identify particular value specifications for human readers, for example elements of a record type.

Table 5.110: ValueSpecification

Class	ArrayValueSpecification			
Package	M2::AUTOSARTemplates::CommonStructure::Constants			
Note	Specifies the values for an array.			
Base	ARObject , ValueSpecification			
Attribute	Datatype	Mul.	Kind	Note

Attribute	Datatype	Mul.	Kind	Note
element (ordered)	ValueSpecification	1..*	aggr	The value for a single array element. All ValueSpecifications aggregated by ArrayValueSpecification shall have the same structure. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime

Table 5.111: ArrayValueSpecification

Class	RecordValueSpecification			
Package	M2::AUTOSARTemplates::CommonStructure::Constants			
Note	Specifies the values for a record.			
Base	ARObject, ValueSpecification			
Attribute	Datatype	Mul.	Kind	Note
field (ordered)	ValueSpecification	1..*	aggr	The value for a single record field. This could also be mapped explicitly to a record element of the data type using the shortName of the ValueSpecification. But this would introduce a relationship to the data type that is too strong. As of now, it is only important that the structure of the data type matches the structure of the ValueSpecification indepenently of the shortNames. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime

Table 5.112: RecordValueSpecification

Class	TextValueSpecification			
Package	M2::AUTOSARTemplates::CommonStructure::Constants			
Note	The purpose of TextValueSpecification is to define the labels that correspond to enumeration values.			
Base	ARObject, ValueSpecification			
Attribute	Datatype	Mul.	Kind	Note
value	VerbatimString	1	ref	This is the value itself. Note that vt uses the operator to separate the values for the different bitfield masks in case that the semantics of the related DataPrototype is described by means of a BITFIELD_TEXTTABLE in the associated CompuMethod.

Table 5.113: TextValueSpecification

Class	NumericalValueSpecification			
Package	M2::AUTOSARTemplates::CommonStructure::Constants			
Note	A numerical ValueSpecification which is intended to be assigned to a Primitive data element. Note that the numerical value is a variant, it can be computed by a formula.			
Base	ARObject, ValueSpecification			
Attribute	Datatype	Mul.	Kind	Note
value	Numerical	1	attr	This is the value itself. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime

Table 5.114: NumericalValueSpecification

Class	ReferenceValueSpecification			
Package	M2::AUTOSARTemplates::CommonStructure::Constants			
Note	Specifies a reference to a data prototype to be used as an initial value for a pointer in the software.			
Base	ARObject, ValueSpecification			
Attribute	Datatype	Mul.	Kind	Note
referenceValue	DataPrototype	1	ref	The referenced data prototype.

Table 5.115: ReferenceValueSpecification

[TPS_SWCT_01178] **Specialized subclasses of [ValueSpecification](#)** [Figure 5.54 shows the specialized subclasses of [ValueSpecification](#) which allow to define values for different use cases:

- Reference to a constant (which is actually a reusable value specification) by means of a [ConstantReference](#)
- [TextValueSpecification](#)
- [NumericalValueSpecification](#)
- [ArrayValueSpecification](#)
- [RecordValueSpecification](#)
- [ApplicationValueSpecification](#): this can be used to specify the value of Compound Primitive Data Types (see [TPS_SWCT_01179]) such as curves and maps. It is also possible to use this in general (e.g. for a primitive calibration value) for the specification of a value of a [DataPrototype](#) typed by an [ApplicationDataType](#) (see section 5.7.4).

Note that [ApplicationValueSpecification](#) is modeled along the example of ASAM CDF (for more information please refer to [27]).

- reference to a [DataPrototype](#): this can be used to describe initial values for pointer variables in the basic software. One use case is the exchange of data de-

scriptions used to access calibration data for software emulation methods (see [7] for details).

- [ApplicationRuleBasedValueSpecification](#)
- [NumericalRuleBasedValueSpecification](#)

]([RS_SWCT_03175](#))

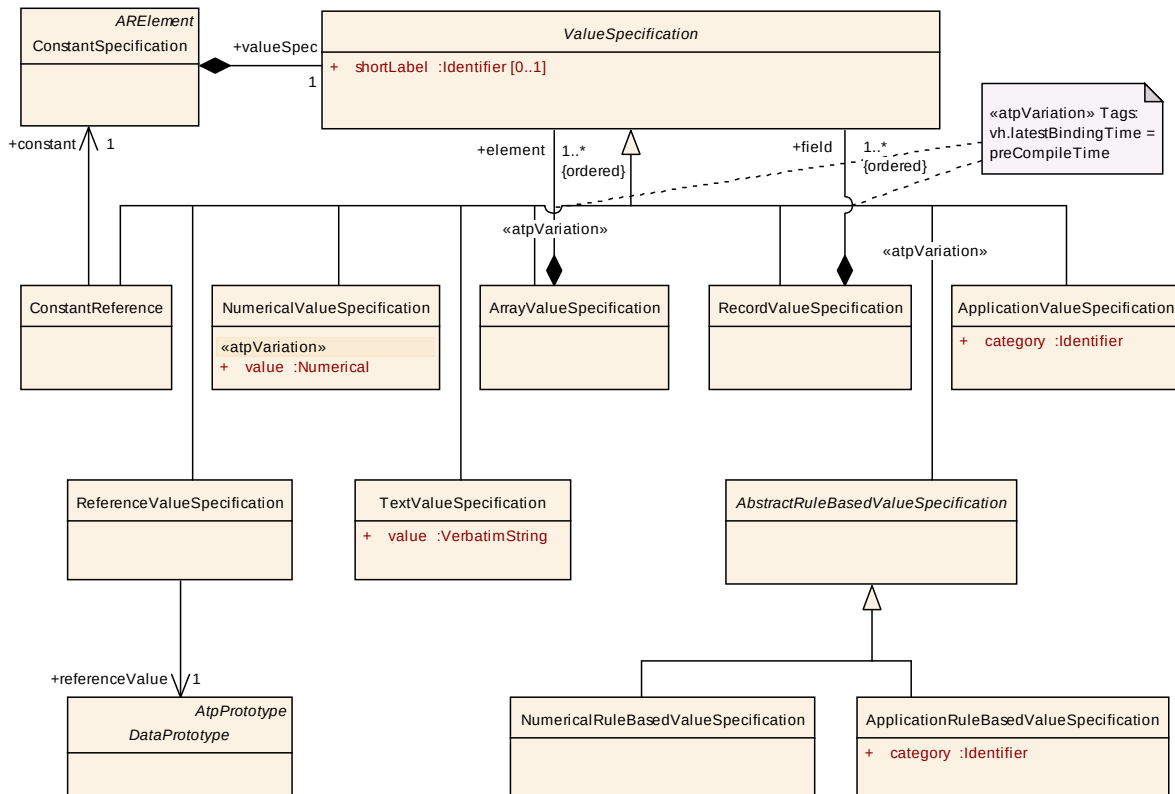


Figure 5.54: Summary of `ValueSpecification`

It is important to understand that although the name of the meta-class `TextValueSpecification` suggests that it is the preferred way for the definition of an `invalidValue` or `initValue` of an `ApplicationPrimitiveDataType` of category `STRING` the `TextValueSpecification` actually has a different purpose (as defined by [constr_1284](#)).

[constr_1284] Limitation of the use of `TextValueSpecification` [`TextValueSpecification` shall **only** be used in the context of an `AutosarDataType` that references a `CompuMethod` in the role `ImplementationDataType.swDataDefPropos.compuMethod` of category `TEXTTABLE`, `BITFIELD_TEXTTABLE`, `SCALE_LINEAR_AND_TEXTTABLE`, and `SCALE_RATIONAL_AND_TEXTTABLE`.]()

In other words, the purpose of `TextValueSpecification` is to define the labels that correspond to enumeration values. The constraints [constr_1225](#) and [constr_1284](#) correspond to each other such that [constr_1225](#) demands the usage of `TextValueSpecification` for the definition of labels for enumeration values while [con-](#)

[str_1284](#)] says that the definition of labels for enumeration values is the only use case for [TextValueSpecification](#).

Note that [ValueSpecification](#) does not inherit from any data type. This would cause a redundancy¹³ in the meta-model since the intended data type of a [ValueSpecification](#) is already determined by the context in which it is aggregated.

Nonetheless the intended data type imposes a certain constraint on the content of a [ValueSpecification](#):

[constr_4035] ValueSpecification shall fit into data type [An instance of [ValueSpecification](#) which is used to assign a value to a software object typed by an [AutosarDataType](#) shall fit into this [AutosarDataType](#) without losing information.]()

For example, it is not allowed to assign the numerical value “1.5” as initial value to a data prototype typed by an [ImplementationDataType](#) which has an integer base type.

[constr_1271] RecordValueSpecification.elements shall be identical to the number of ApplicationRecordDataType.element [The initialization of an [DataPrototype](#) typed by an [ApplicationRecordDataType](#) by means of a [RecordValueSpecification](#) shall exactly match the structure of the [ApplicationRecordDataType](#).]()

For this means, it is required that the number of [RecordValueSpecification.elements](#) shall be identical to the number of [ApplicationRecordDataType.elements](#).]()

[constr_1272] RecordValueSpecification.elements shall be identical to the number of subElements of ImplementationDataType of category STRUCTURE [The initialization of an [DataPrototype](#) typed by an [ImplementationDataType](#) of category [STRUCTURE](#) by means of a [RecordValueSpecification](#) shall exactly match the structure of the [ImplementationDataType](#) of category [STRUCTURE](#).]()

For this means, it is required that the number of [RecordValueSpecification.elements](#) shall be identical to the number of [ImplementationDataType.subElements](#).]()

[constr_1273] ArrayValueSpecification.elements shall be identical to the value of ApplicationArrayType.element.maxNumberOfElements [The initialization of [DataPrototype](#) typed by an [ApplicationArrayType](#) by means of an [ArrayValueSpecification](#) shall exactly match the structure of the [ApplicationArrayType](#) regardless of the setting of the attribute [ApplicationArrayType.element.arraySizeSemantics](#).]()

¹³For example, “1” can be taken as a constant value for many data types. If the [ValueSpecification](#) would refer to a specific [AutosarDataType](#) it would be necessary to define a “1” for every single [AutosarDataType](#) this value is supposed to be used in combination with.

This means that the number of `ArrayValueSpecification.elements` shall be identical to the value of `ApplicationArrayDataType.element.maxNumberOfElements`. `]()`

The consequence of `[constr_1273]` is that for e.g. an `ApplicationArrayDataType` that has the attribute `element.maxNumberOfElements` set to the value 10 a `ArrayValueSpecification` shall be provided that contains 10 elements.

[constr_1274] `ArrayValueSpecification.elements` shall be identical to the value of `ImplementationDataType.subElement.arraySize` of category **ARRAY** `[` The initialization of a `DataPrototype` typed by an `ImplementationDataType` of category `ARRAY` by means of an `ArrayValueSpecification` shall exactly match the structure of the `ImplementationDataType` regardless of the setting of the attribute `ImplementationDataType.subElement.arraySizeSemantics`. `]`

This means that the number of `ArrayValueSpecification.elements` shall be identical to the value of `ImplementationDataType.subElement.arraySize`. `]()`

For deeply nested composite data types (including `ImplementationDataTypes` created in response to the existence of a Compound Primitive Data Type) `[constr_1271]`, `[constr_1272]`, and `[constr_1273]` shall be applied recursively according to the nature of the given nesting levels. For the “leaf” elements `[constr_4035]` applies.

5.6.2 Specification of Values based on Rules

[TPS_SWCT_01484] Meaning of `ApplicationRuleBasedValueSpecification` `[` The purpose of the `ApplicationRuleBasedValueSpecification` is to provide means for a compact provision of values for `DataPrototypes` that otherwise would require a high volume (in terms of serialized ARXML) of e.g. initialization data. `ApplicationRuleBasedValueSpecification` may be used for `ApplicationArrayDataType`, and also (if applicable) to the so-called Compound Primitive Data Types. `]` ([RS_SWCT_03260](#))

For example, an `ApplicationArrayDataType` that has 100 elements would need to be initialized such that for each element a dedicated initial value is provided. In the most prominent cases the majority of these elements are initialized with an identical value (e.g. 0) and only the first few elements differ in terms of initialization values.

Class	AbstractRuleBasedValueSpecification (abstract)			
Package	M2::AUTOSARTemplates::CommonStructure::Constants			
Note	This represents an abstract base class for all rule-based value specifications.			
Base	ARObject, ValueSpecification			
Attribute	Datatype	Mul.	Kind	Note
—	—	—	—	—

Table 5.116: AbstractRuleBasedValueSpecification

Class	ApplicationRuleBasedValueSpecification			
Package	M2::AUTOSARTemplates::CommonStructure::Constants			
Note	This meta-class represents rule based values for DataPrototypes typed by ApplicationDataTypes (ApplicationArrayDataType or a compound ApplicationPrimitiveDataType which also boils down to an array-nature).			
Base	ARObject, AbstractRuleBasedValueSpecification , ValueSpecification			
Attribute	Datatype	Mul.	Kind	Note
category	Identifier	1	ref	This represents the category of the RuleBasedValueSpecification Tags: xml.sequenceOffset=-20
swAxisCont (ordered)	RuleBasedAxisCont	*	aggr	This represents the axis values of a Compound Primitive Data Type (curve or map). The first swAxisCont describes the x-axis, the second swAxisCont describes the y-axis, the third swAxisCont describes the z-axis. In addition to this, the axis can be denoted in swAxisIndex.
swValueCont	RuleBasedValueCont	0..1	aggr	This represents the values of an array or Compound Primitive Data Type.

Table 5.117: ApplicationRuleBasedValueSpecification

Class	RuleBasedAxisCont			
Package	M2::AUTOSARTemplates::CommonStructure::Constants			
Note	This represents the values for the axis of a compound primitive (curve, map). For standard and fix axes, SwAxisCont contains the values of the axis directly. The axis values of SwAxisCont with the category CURVE_AXIS, COM_AXIS, RES_AXIS are for display only. For editing and processing, only the values in the related GroupAxis are binding.			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
category	CalprmAxisCategoryEnum	1	attr	This category specifies the particular axis types: • FIX_AXIS • STD_AXIS • COM_AXIS • CURVE_AXIS (swArraysize necessary) • RES_AXIS (swArraysize necessary) Tags: xml.sequenceOffset=20
ruleBasedValues	RuleBasedValueSpecification	1	aggr	This represents the rule based value specification for the axis of a compound primitive (curve, map). Tags: xml.roleElement=true; xml.roleWrapperElement=false; xml.sequenceOffset=80; xml.typeWrapperElement=false

Attribute	Datatype	Mul.	Kind	Note
swArraysize	ValueList	1	aggr	For multidimensional compound primitives (curve, map ...) it is necessary to know the dimensions. They are specified using swArraySize. Tags: xml.sequenceOffset=40
swAxisIndex	AxisIndexType	1	attr	This property allows to explicitly assign the axis contents to a particular axis. It is specified by numbers where 1 corresponds to the x-axis. It is also possible to derive the axis association from the sequence of the parent. Tags: xml.sequenceOffset=50
unit	Unit	0..1	ref	This represents the physical unit of the provided values. Tags: xml.sequenceOffset=30

Table 5.118: RuleBasedAxisCont

Class	RuleBasedValueCont			
Package	M2::AUTOSARTemplates::CommonStructure::Constants			
Note	This represents the values of a compound primitive (CURVE, MAP, CUBOID, CUBE_4, CUBE_5, VAL_BLK) or an array.			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
ruleBasedValues	RuleBasedValueSpecification	1	aggr	This represents the rule based value specification for the array or compound primitive (CURVE, MAP, CUBOID, CUBE_4, CUBE_5, VAL_BLK). Tags: xml.roleElement=true; xml.roleWrapperElement=false; xml.sequenceOffset=80; xml.typeWrapperElement=false
swArraysize	ValueList	0..1	aggr	This attribute defines the size of each dimension for compound primitives CURVE, MAP, CUBOID, CUBE_4, CUBE_5, COM_AXIS, RES_AXIS, CURVE_AXIS, VAL_BLK, STRING. For each dimension one value has to be defined, e.g. one in case of COM_AXIS and two or more in case of MAP. Tags: xml.sequenceOffset=40
unit	Unit	0..1	ref	This represents the physical unit of the provided values. Tags: xml.sequenceOffset=30

Table 5.119: RuleBasedValueCont

In case the [ApplicationRuleBasedValueSpecification](#) is applied to Compound Primitive Data Types basically the same rules apply for [Application-](#)

`RuleBasedValueSpecification` as defined for `ApplicationValueSpecification`.

[constr_2057] Mandatory information of a `RuleBasedAxisCont` [If the attribute `swAxisCont` is defined for an `ApplicationRuleBasedValueSpecification` the `RuleBasedAxisCont` shall define one `swAxisIndex` value and one `swArraysize` value per dimension, even in the case when the owning `ApplicationRuleBasedValueSpecification` defines only the content of a single dimensional object like a `CURVE`.]()

[constr_2058] Mandatory information of a `RuleBasedValueCont` [If the attribute `swValueCont` is defined for an `ApplicationRuleBasedValueSpecification` the `RuleBasedValueCont` shall define always the attribute `swArraysize` if the `ApplicationRuleBasedValueSpecification` is of category `CURVE`, `MAP`, `CUBOID`, `CUBE_4`, `CUBE_5`, `COM_AXIS`, `RES_AXIS`, `CURVE_AXIS`, `VAL_BLK` or `ARRAY`.]()

Please note that for multidimensional `Compound Primitive Data Types` (e.g. `MAP`) it is necessary to know the dimensions in order to be able to process the `SwValues`. **[constr_2057]** and **[constr_2058]** shall support a consistent handling of single and multidimensional `Compound Primitive Data Types`.

If the `ApplicationRuleBasedValueSpecification` defines values for a `Compound Primitive Data Type` with more than one input axis the `swArraySize` gets mandatory to ensure the correct processing of the values calculated by rule.

[TPS_SWCT_02053] Values of `RuleBasedAxisCont` with the category `CURVE_AXIS`, `COM_AXIS`, `RES_AXIS` are for display only [In case of `ApplicationRuleBasedValueSpecifications` of category `MAP`, `CUBOID`, `CUBE_4`, `CUBE_5` or `CURVE` it is possible that the `RuleBasedAxisCont` of axes can be omitted if the axis is of category `COM_AXIS` or `RES_AXIS` or `CURVE_AXIS`.

If `RuleBasedAxisCont` values exists in such cases for the axes these are for display purpose only because the related `DataPrototype` of the `MAP` or `CURVE` does not hold the values of such axes. These are properties of the `DataPrototype` of the `COM_AXIS` or `RES_AXIS` or `CURVE_AXIS`.]()

Hence, values of the `COM_AXIS` itself are described by `RuleBasedValueCont`.

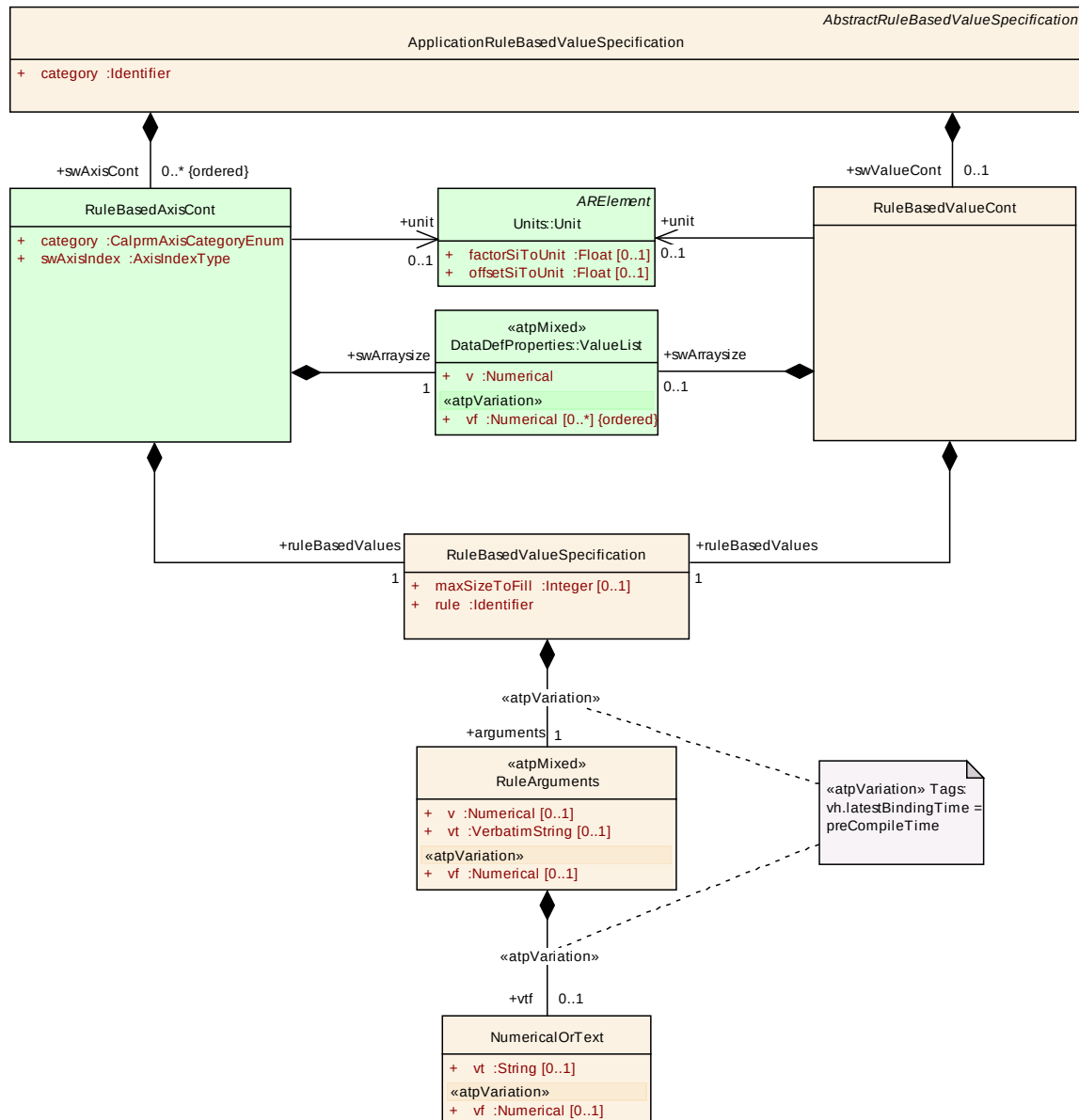


Figure 5.55: Definition of an `ApplicationRuleBasedValueSpecification`

[TPS_SWCT_01528] Meaning of `NumericalRuleBasedValueSpecification` [The purpose of the `NumericalRuleBasedValueSpecification` is to provide means for a compact provision of values for `DataPrototypes` that otherwise would require a high volume (in terms of serialized ARXML) of e.g. initialization data. `NumericalRuleBasedValueSpecification` may be used for `DataPrototypes` typed by `ImplementationDataTypes` of category `ARRAY` or Compound Primitive Data Types mapped to `ImplementationDataTypes` of category `ARRAY`.] ([RS_SWCT_03260](#))

Concerning `initValues` for Compound Primitive Data Types please note as well [TPS_SWCT_01185].

Class	NumericalRuleBasedValueSpecification			
Package	M2::AUTOSARTemplates::CommonStructure::Constants			
Note	This meta-class is used to support a rule-based initialization approach for data types with an array-nature (ImplementationDataType of category ARRAY).			
Base	ARObject, AbstractRuleBasedValueSpecification , ValueSpecification			
Attribute	Datatype	Mul.	Kind	Note
ruleBased Values	RuleBasedValueSpecification	1	aggr	This represents the rule based value specification for the array. Tags: xml.roleElement=true; xml.roleWrapperElement=false; xml.typeWrapperElement=false

Table 5.120: NumericalRuleBasedValueSpecification

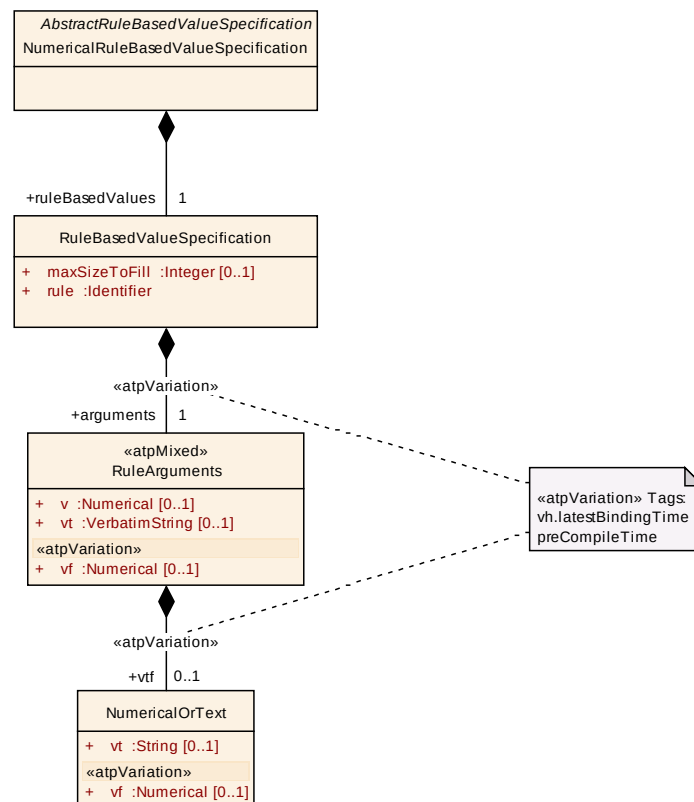


Figure 5.56: Definition of an [NumericalRuleBasedValueSpecification](#)

[TPS_SWCT_01495] Standardized value of [RuleBasedValueSpecification.rule](#) [AUTOSAR reserves a dedicated value of [RuleBasedValueSpecification.rule](#) in a standardized semantics:

- FILL_UNTIL_END
- FILL_UNTIL_MAX_SIZE

The meaning of this value of [rule](#) is explained in [TPS_SWCT_01494] and [TPS_SWCT_01609].]([RS_SWCT_03260](#), [RS_SWCT_03181](#))

[TPS_SWCT_01485] The order of `RuleArguments` arguments shall be respected
 [The order of arguments in `RuleArguments` corresponds to the order of elements in the array, i.e. the first argument corresponds to the first element of the array, the second argument corresponds to the second element of the array, and so on.]([RS_SWCT_03260](#))

Please note that a single argument can be defined by the attributes

- `RuleArguments.v`
- `RuleArguments.vf`
- `RuleArguments.vt`
- `RuleArguments.vtf.vf`
- `RuleArguments.vtf.vt`

[TPS_SWCT_01493] The number of `RuleArguments.arguments` shall not exceed the array size
 [If the number of `RuleArguments.arguments` exceeds the number of elements of an array that it is applied to then the `RuleArguments.arguments` that go beyond the last element of the array shall be ignored.]([RS_SWCT_03260](#))

[TPS_SWCT_01494] A `RuleBasedValueSpecification` of rule `FILL_UNTIL_END` shall fill the value of the last `RuleArguments.argument` until the last element of the array
 [The following rule applies to `RuleBasedValueSpecifications` of rule `FILL_UNTIL_END`:

If the number of `RuleArguments.arguments` is smaller than the number of elements of the array it is applied to then the value of the last `RuleArguments.argument` shall be applied to any following element of the array until the last element of the array.]([RS_SWCT_03260](#))

[TPS_SWCT_01609] A `RuleBasedValueSpecification` of rule `FILL_UNTIL_MAX_SIZE` shall fill the value of the last `RuleArguments.argument` until the number of elements specified in `maxSizeToFill`
 [The following rule applies to `RuleBasedValueSpecifications` of rule `FILL_UNTIL_MAX_SIZE`:

If the number of `RuleArguments.arguments` is smaller than the number of elements of the array it is applied to and smaller than `maxSizeToFill`, then the value of the last `RuleArguments.argument` shall be applied to so many of the following elements that the first `maxSizeToFill` elements of the array are filled.]([RS_SWCT_03260](#))

Class	RuleBasedValueSpecification			
Package	M2::AUTOSARTemplates::CommonStructure::Constants			
Note	This meta-class is used to support a rule-based initialization approach for data types with an array-nature (ApplicationArrayDataType and ImplementationDataType of category ARRAY) or a compound ApplicationPrimitiveDataType (which also boils down to an array-nature).			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note

Attribute	Datatype	Mul.	Kind	Note
arguments	RuleArguments	1	aggr	This represents the arguments for the RuleBasedValueSpecification. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime xml.sequenceOffset=30
maxSizeTo Fill	Integer	0..1	attr	If a rule is chosen which does not fill until the end, this determines until which size the rule shall fill the values. Tags: xml.sequenceOffset=40
rule	Identifier	1	ref	This denotes the name of the rule of the RuleBasedValueSpecification. The rule determines the calculation specification according which the arguments are used to calculated the values. Tags: xml.sequenceOffset=20

Table 5.121: RuleBasedValueSpecification

Class	«atpMixed» RuleArguments			
Package	M2::AUTOSARTemplates::CommonStructure::Constants			
Note	This represents the arguments for a rule-based value specification.			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
v	Numerical	0..1	attr	This represents a numerical value for the RuleBasedValueSpecification.
vf	Numerical	0..1	attr	This represents a numerical value for the RuleBasedValueSpecification which may subject to variability. The latest binding time of the VariationPoint shall be preCompileTime. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime
vt	VerbatimString	0..1	ref	This represents a textual value for the RuleBasedValueSpecification.
vtf	NumericalOrText	0..1	aggr	This aggregation represents the ability to provide a value that is either numerical or text which existence is subject to variability. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime

Table 5.122: RuleArguments

5.6.3 Reference to Constant

Note the specific meaning of [ConstantReference](#): it passes the definition of the value on to a [ConstantSpecification](#) that is defined as part of an AUTOSAR [ARPackage](#).

Class	ConstantReference			
Package	M2::AUTOSARTemplates::CommonStructure::Constants			
Note	Instead of defining this value inline, a constant is referenced.			
Base	ARObject, ValueSpecification			
Attribute	Datatype	Mul.	Kind	Note
constant	ConstantSpecification	1	ref	The referenced constant.

Table 5.123: ConstantReference

5.6.4 Values for Compound Primitive Data Types

[TPS_SWCT_01180] Maximum possible size of Compound Primitive Data Type [Note that if the size of the Compound Primitive Data Type (see [\[TPS_SWCT_01179\]](#)) (curve/map) is defined using an [AttributeValueVariationPoint](#) (in other words [swMaxAxisPoints](#), [swValueBlockSize](#) dependent on the value of [SwSystemconst](#)) the `initValue` shall provide the maximum possible amount of values.]([RS_SWCT_03216](#))

In this case it is the responsibility of model author to ensure that the size of the specified init values matches the range of the involved system constants.

[constr_1160] Size of Compound Primitive Data Type is variant [For Compound Primitive Data Types (see [\[TPS_SWCT_01179\]](#)) where the size is subject to variation the size of the specified `initValues` shall match the range of the involved [SwSystemconst](#).]()

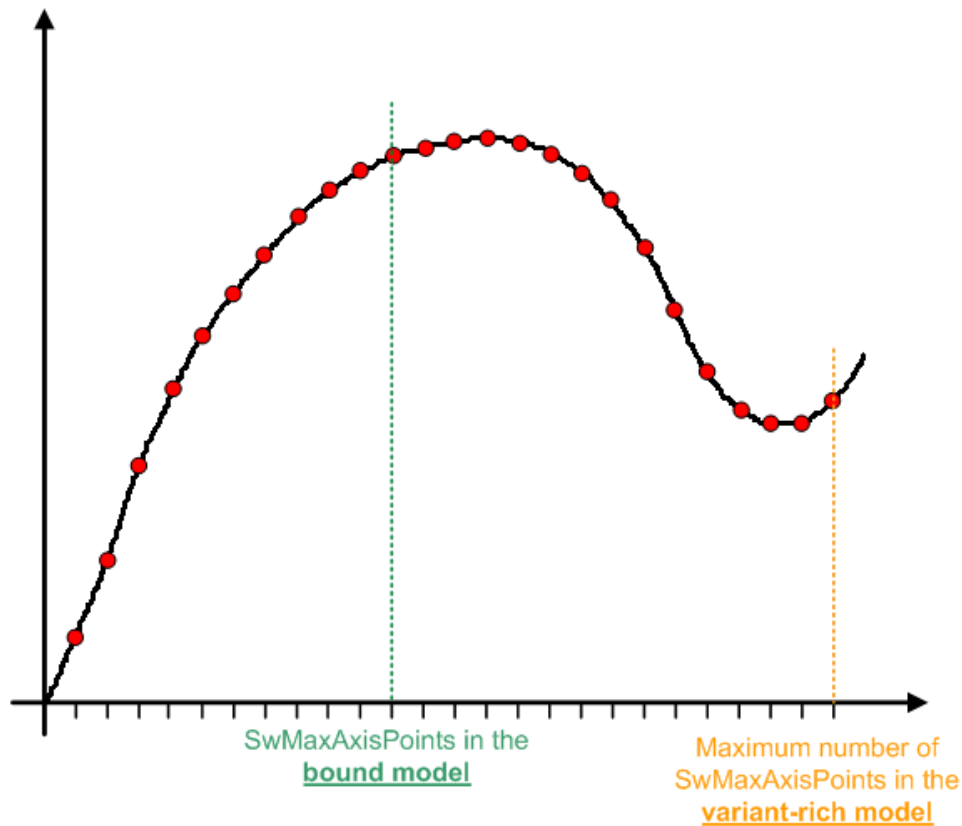


Figure 5.57: Explanation of **swMaxAxisPoints**

[TPS_SWCT_01181] **Bound model specifies a primitive which is smaller than the maximum defined by the range of the involved **SwSystemconst**** [The processing tools shall take the lower part of the `initValues` in case the bound model specifies a primitive which is smaller than the maximum defined by the range of the involved `SwSystemconst`.]([RS_SWCT_03216](#), [RS_SWCT_03148](#))

The consequences of [TPS_SWCT_01181] are exemplified by Figure 5.57.

[constr_2050] **Mandatory information of a **SwAxisCont**** [If the attribute `swAxisCont` is defined for an `ApplicationValueSpecification` the `SwAxisCont` shall define one `swAxisIndex` value and one `swArraysize` value per dimension, even in the case when the owning `ApplicationValueSpecification` defines only the content of a single dimensional object like a CURVE.]()

[constr_2051] **Mandatory information of a **SwValueCont**** [If the attribute `swValueCont` is defined for an `ApplicationValueSpecification` the `SwValueCont` shall always define the attribute `swArraysize` if the `ApplicationValueSpecification` is of category CURVE, MAP, CUBOID, CUBE_4, CUBE_5, COM_AXIS, RES_AXIS, CURVE_AXIS, or VAL_BLK.]()

Please note that for multidimensional Compound Primitive Types (e.g. MAP) it is necessary to know the dimensions in order to be able to process the `SwValues`. [constr_2050] and [constr_2051] shall support a consistent handling of single and multidimensional Compound Primitive Data Types.

[constr_2052] Values of `swArraySize` and the number of values provided by `swValuesPhys` shall be consistent. [`swValuesPhys` shall define as many numbers of values as the `swArraySize` defines. In other words, in the bound model the number of descendants (`v`, or `vf`, or `vt`, or `vtf`) shall be identical to the number of elements of the related `DataPrototype` typed by an `ApplicationPrimitiveDataType`.

If several `swArraySize` values are provided these have to be multiplied in order to get the total number of `swValuesPhys` values. `]()`

Please note that case of Compound Primitive Data Types typically the attribute `swValuesPhys` defines more than one value. [constr_2051] and [constr_2052] shall enable a consistent handling of the `swValuesPhys` values regardless how many dimensions the related Compound Primitive Type defines.

If the `ApplicationValueSpecification` defines values for a Compound Primitive Data Type with more than one input axis the `swArraySize` gets mandatory to ensure the correct processing of the `swValuesPhys` values independent of the existence of `SwValues.vg`.

[TPS_SWCT_02001] Values of `SwAxisCont` with the category `CURVE_AXIS`, `COM_AXIS`, `RES_AXIS` are for display only [In case of `ApplicationValueSpecifications` of category `MAP`, `CUBOID`, `CUBE_4`, `CUBE_5`, and `CURVE` it is possible that the `SwAxisCont` of axes can be omitted if the axis is of category `COM_AXIS` or `RES_AXIS` or `CURVE_AXIS`.

If `SwAxisCont` values exists in such cases for the axes these are for display purpose only because the related `DataPrototype` of the `MAP`, `CUBOID`, `CUBE_4`, `CUBE_5`, or `CURVE` does not hold the values of such axes. These are properties of the `DataPrototype` of the `COM_AXIS` or `RES_AXIS` or `CURVE_AXIS`. `]()`

Hence values of the `COM_AXIS` itself are described by `SwValueCont`.

[constr_1243] `NumericalOrText` shall either define `vf` or `vt` [Within the context of one `NumericalOrText`, either the attribute `vf` or the attribute `vt` shall be defined. The existence of both attributes at the same time is not permitted. `]()`

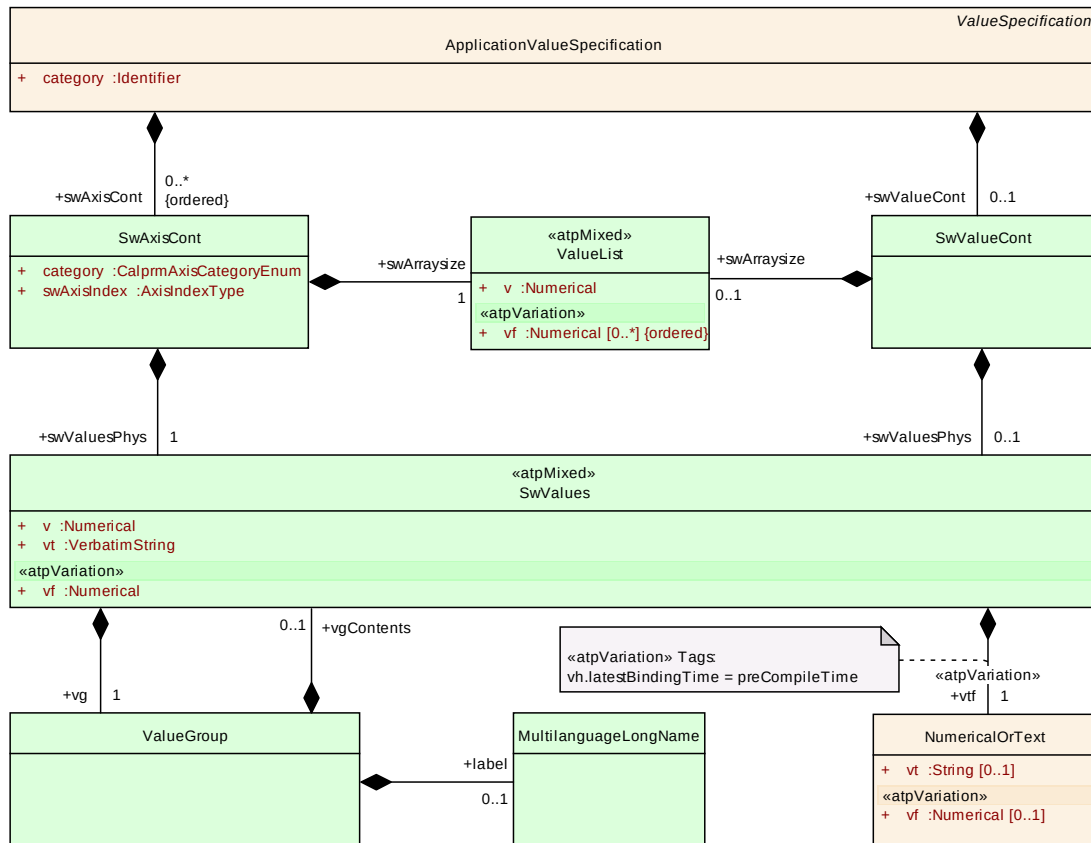


Figure 5.58: Definition of an `ApplicationValueSpecification`

Class	ApplicationValueSpecification			
Package	M2::AUTOSARTemplates::CommonStructure::Constants			
Note	This meta-class represents values for DataPrototypes typed by ApplicationDataTypes (this includes in particular compound primitives). For further details refer to ASAM CDF 2.0. This meta-class corresponds to some extent with SW-INSTANCE in ASAM CDF 2.0.			
Base	ARObject, ValueSpecification			
Attribute	Datatype	Mul.	Kind	Note
category	Identifier	1	ref	Specifies to which category of ApplicationDataType this ApplicationValueSpecification can be applied (e.g. as an initial value), thus imposing constraints on the structure and semantics of the contained values, see [constr_1006] and [constr_2051] .
swAxisCont (ordered)	SwAxisCont	*	aggr	This represents the axis values of a Compound Primitive Data Type (curve or map). The first swAxisCont describes the x-axis, the second swAxisCont describes the y-axis, the third swAxisCont describes the z-axis. In addition to this, the axis can be denoted in swAxisIndex.
swValueCont	SwValueCont	0..1	aggr	This represents the values of a Compound Primitive Data Type.

Attribute	Datatype	Mul.	Kind	Note

Table 5.124: ApplicationValueSpecification

Class	SwAxisCont			
Package	M2::AUTOSARTemplates::CommonStructure::CalibrationValue			
Note	This represents the values for the axis of a compound primitive (curve, map). For standard and fix axes, SwAxisCont contains the values of the axis directly. The axis values of SwAxisCont with the category CURVE_AXIS, COM_AXIS, RES_AXIS are for display only. For editing and processing, only the values in the related GroupAxis are binding.			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
category	CalprmAxisCategoryEnum	1	attr	This category specifies the particular axis types: • FIX_AXIS • STD_AXIS • COM_AXIS • CURVE_AXIS (swArraysize necessary) • RES_AXIS (swArraysize necessary) Tags: xml.sequenceOffset=20
swArraysize	ValueList	1	aggr	For multidimensional compound primitives (curve, map ...) it is necessary to know the dimensions. They are specified using swArraySize. • RES_AXIS • CURVE_AXIS Tags: xml.sequenceOffset=70
swAxisIndex	AxisIndexType	1	attr	This property allows to explicitly assign the axis contents to a particular axis. It is specified by numbers where 1 corresponds to the x-axis. It is also possible to derive the axis association from the sequence of the parent. Tags: xml.sequenceOffset=50
swValuesPhys	SwValues	1	aggr	swValuesPhys represents the values in the physical domain. Tags: xml.sequenceOffset=80
unit	Unit	1	ref	This represents the physical unit of the provided values. Tags: xml.sequenceOffset=30

Attribute	Datatype	Mul.	Kind	Note
unitDisplay Name	SingleLanguage UnitNames	0..1	aggr	This represents the display name which is used for the physical unit of the axis. Tags: xml.sequenceOffset=40

Table 5.125: SwAxisCont

Class	SwValueCont			
Package	M2::AUTOSARTemplates::CommonStructure::CalibrationValue			
Note	This metaclass represents the content of one particular SwInstance.			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
swArraysiz e	ValueList	0..1	aggr	This attribute defines the size of each dimension for compound primitives CURVE, MAP, CUBOID, CUB_4, CUBE_5, COM_AXIS, RES_AXIS, CURVE_AXIS, VAL_BLK, STRING. For each dimension one value has to be defined, e.g. one in case of COM_AXIS and two or more in case of MAP. Tags: xml.sequenceOffset=40
swValuesP hys	SwValues	0..1	aggr	swValuesPhys represents the values in the physical domain. Tags: xml.sequenceOffset=50
unit	Unit	1	ref	This represents the physical unit of the provided values. Tags: xml.sequenceOffset=20
unitDisplay Name	SingleLanguage UnitNames	0..1	aggr	This specifies how the physical units of the current value set shall be displayed in documents or in user interfaces of tools. Tags: xml.sequenceOffset=30

Table 5.126: SwValueCont

Class	«atpMixed» SwValues			
Package	M2::AUTOSARTemplates::CommonStructure::CalibrationValue			
Note	This meta-class represents a list of values. These values can either be the input values of a curve (abscissa values) or the associated values (ordinate values). In case of multidimensional structures, the values are ordered such that the lowest index runs the fastest. In particular for maps and cuboids etc. the resulting long value list can be subsectioned using ValueGroup. But the processing needs to be done as if vg is not there. Note that numerical values and textual values should not be mixed.			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
v	Numerical	1	attr	This is a non variant Value. It is provided for sake of Compatibility to ASAM CDF. Tags: xml.sequenceOffset=40
vf	Numerical	1	attr	This allows to specify the value as VariationPoint. It is distinguished to non variant for sake of compatibility to ASAM CDF 2.0. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime xml.sequenceOffset=20
vg	ValueGroup	1	aggr	This allows to have intersections in the values in order to support specific rendering (eg. using stylesheets). For tools it is important that the v values are always processed in the same (flattened) order and the tool is able to interpret it without respecting vg. Tags: xml.sequenceOffset=50
vt	VerbatimString	1	ref	This represents the values of textual data elements (Strings). Note that vt uses the to separate the values for the different bitfield masks in case that the semantics of the related DataPrototype is described by means of a BITFIELD_TEXTTABLE in the associated CompuMethod. Tags: xml.sequenceOffset=30
vtf	NumericalOrText	1	aggr	This aggregation represents the ability to provide a value that is either numerical or text which existence is subject to variability. From the formal point of view, the aggregation needs to have the multiplicity 1 because SwValues is modelled with stereotype «atpMixed». Nevertheless, the existence of vtf is optional and subject to constraints. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime

Attribute	Datatype	Mul.	Kind	Note

Table 5.127: SwValues

Class	ValueGroup			
Package	M2::AUTOSARTemplates::CommonStructure::CalibrationValue			
Note	This element enables values to be grouped. It can be used to perform row and column-orientated groupings, so that these can be rendered properly e.g. as a table.			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
label	MultilanguageLongName	0..1	aggr	This label allows to give the valueGroup a particular name. It can be used if the Values are rendered as a table. Tags: xml.sequenceOffset=20
vgContents	SwValues	0..1	aggr	This represents the contents of the value group. Tags: xml.roleElement=false; xml.roleWrapperElement=false; xml.sequenceOffset=30; xml.typeElement=false; xml.typeWrapperElement=false

Table 5.128: ValueGroup

Class	«atpMixed» ValueList			
Package	M2::AUTOSARTemplates::CommonStructure::DataDefProperties			
Note	This is a generic list of numerical values.			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
v	Numerical	1	attr	This is a particular numerical value without variation. Tags: xml.sequenceOffset=30
vf (ordered)	Numerical	*	attr	This is one entry in the list of numerical values Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime xml.roleElement=true; xml.roleWrapperElement=false; xml.typeElement=false; xml.typeWrapperElement=false

Table 5.129: ValueList

Class	NumericalOrText			
Package	M2::AUTOSARTemplates::CommonStructure::Constants			
Note	This meta-class represents the ability to yield either a numerical or a string. A typical use case is that two or more instances of this meta-class are aggregated with a VariationPoint where some instances yield strings while other instances yield numerical depending on the resolution of the binding expression.			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
vf	Numerical	0..1	attr	This attribute represents the ability to provide a numerical value. The latest binding time of the VariationPoint shall be preCompileTime. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime xml.sequenceOffset=10
vt	String	0..1	attr	This attribute represents the ability to provide a textual value. Tags: xml.sequenceOffset=20

Table 5.130: NumericalOrText

5.6.5 Examples

5.6.5.1 Example for Constant Specification for CURVE

The following example illustrates how a [ConstantSpecification](#) is specified for a [CURVE](#). Please note, that in this example the [vf](#) attribute is used for the [swArraysize](#) as well as for the [swValuesPhys](#). The basic intention of [vf](#) is the usage for variant rich models but it is valid as well if [vf](#) contains invariant values.

Listing 5.13: Example for Constant Specification for CURVE

```

<CONSTANT-SPECIFICATION>
  <SHORT-NAME>PhysInitValuesOfCurve</SHORT-NAME>
  <DESC>
    <L-2 L="EN">This example shows a ConstantSpecification for a
      CURVE where the axis is a STD_AXIS</L-2>
  </DESC>
  <VALUE-SPEC>
    <APPLICATION-VALUE-SPECIFICATION>
      <CATEGORY>CURVE</CATEGORY>
      <SW-AXIS-CONTS>
        <SW-AXIS-CONT>
          <CATEGORY>STD_AXIS</CATEGORY>
          <SW-AXIS-INDEX>1</SW-AXIS-INDEX>
          <SW-ARRAYSIZE>
            <VF>4</VF>
          </SW-ARRAYSIZE>
          <SW-VALUES-PHYS>
            <VF>0</VF>
            <VF>1</VF>
          </SW-VALUES-PHYS>
        </SW-AXIS-CONT>
      </SW-AXIS-CONTS>
    </APPLICATION-VALUE-SPECIFICATION>
  </VALUE-SPEC>
</CONSTANT-SPECIFICATION>

```

```

        <VF>2</VF>
        <VF>3</VF>
    </SW-VALUES-PHYS>
</SW-AXIS-CONT>
</SW-AXIS-CONTS>
<SW-VALUE-CONT>
    <UNIT-REF DEST="UNIT">/AUTOSAR/AISpecification/Units/NwtMtr</
        UNIT-REF>
    <SW-ARRAYSIZE>
        <VF>4</VF>
    </SW-ARRAYSIZE>
    <SW-VALUES-PHYS>
        <VF>00.000</VF>
        <VF>10.000</VF>
        <VF>20.000</VF>
        <VF>30.000</VF>
    </SW-VALUES-PHYS>
</SW-VALUE-CONT>
</APPLICATION-VALUE-SPECIFICATION>
</VALUE-SPEC>
</CONSTANT-SPECIFICATION>

```

5.6.5.2 Example for Constant Specification for MAP

The following example illustrates how an `ConstantSpecification` is specified for a MAP. In this case one axis of the MAP is a `STD_AXIS` and the second one is a `COM_AXIS`. Please note that in this example the `v` attribute is used for the `swArraysize` as well as for the `swValuesPhys`. This is possible because the example contains only invariant values.

Listing 5.14: Example for Constant Specification for MAP

```

<CONSTANT-SPECIFICATION>
  <SHORT-NAME>PhysInitValuesOfMap</SHORT-NAME>
  <DESC>
    <L-2 L="EN">This example shows a ConstantSpecification for a MAP
      where the first axis is a STD_AXIS and the second axis is a
      COM_AXIS</L-2>
  </DESC>
  <VALUE-SPEC>
    <APPLICATION-VALUE-SPECIFICATION>
      <CATEGORY>MAP</CATEGORY>
      <SW-AXIS-CONTS>
        <SW-AXIS-CONT>
          <CATEGORY>STD_AXIS</CATEGORY>
          <SW-AXIS-INDEX>1</SW-AXIS-INDEX>
          <SW-ARRAYSIZE>
            <V>4</V>
          </SW-ARRAYSIZE>
          <SW-VALUES-PHYS>
            <V>0</V>
            <V>1</V>
            <V>2</V>

```

```

        <V>3</V>
    </SW-VALUES-PHYS>
</SW-AXIS-CONT>
</SW-AXIS-CONTS>
<SW-VALUE-CONT>
    <UNIT-REF DEST="UNIT">/AUTOSAR/AISpecification/Units/NwtMtr</
        UNIT-REF>
    <SW-ARRAYSIZE>
        <V>4</V>
        <V>2</V>
    </SW-ARRAYSIZE>
    <SW-VALUES-PHYS>
        <VG>
            <LABEL>
                <L-4 L="EN">Values for axis index 2 equals 0</L-4>
            </LABEL>
            <V>00</V>
            <V>10</V>
            <V>20</V>
            <V>30</V>
        </VG>
        <VG>
            <LABEL>
                <L-4 L="EN">Values for axis index 2 equals 1</L-4>
            </LABEL>
            <V>01</V>
            <V>11</V>
            <V>21</V>
            <V>31</V>
        </VG>
    </SW-VALUES-PHYS>
</SW-VALUE-CONT>
</APPLICATION-VALUE-SPECIFICATION>
</VALUE-SPEC>
</CONSTANT-SPECIFICATION>

```

5.6.5.3 Example for Constant Specification for COM_AXIS

The following example illustrates how an [ConstantSpecification](#) is specified for a [COM_AXIS](#).

Listing 5.15: Example for Constant Specification for COM_AXIS

```

<CONSTANT-SPECIFICATION>
    <SHORT-NAME>PhysInitValuesOfComAxis</SHORT-NAME>
    <DESC>
        <L-2 L="EN">This example shows a ConstantSpecification for a
            COM_AXIS</L-2>
    </DESC>
    <VALUE-SPEC>
        <APPLICATION-VALUE-SPECIFICATION>
            <CATEGORY>COM_AXIS</CATEGORY>
            <SW-VALUE-CONT>

```

```

<UNIT-REF DEST="UNIT">/AUTOSAR/AISpecification/Units/Rpm</UNIT-
REF>
<SW-ARRAYSIZE>
  <V>6</V>
</SW-ARRAYSIZE>
<SW-VALUES-PHYS>
  <V>0</V>
  <V>500</V>
  <V>1000</V>
  <V>1500</V>
  <V>3000</V>
  <V>5000</V>
</SW-VALUES-PHYS>
</SW-VALUE-CONT>
</APPLICATION-VALUE-SPECIFICATION>
</VALUE-SPEC>
</CONSTANT-SPECIFICATION>

```

5.7 Initial Values

5.7.1 Overview

[TPS_SWCT_01301] Importance of initial values [If the value of a [VariableDataPrototype/ParameterDataPrototype](#) has not properly been set by a piece of software it can still happen that another piece of software tries to access the value of the [VariableDataPrototype/ParameterDataPrototype](#).

For various reasons it is therefore advised to be able to specify an initial value for a [VariableDataPrototype/ParameterDataPrototype](#) in case the value has not been assigned in a controlled manner. However, the definition of an initial value in many cases depends on a context in which the value is accessed.]()

Therefore, the AUTOSAR standard foresees means for defining initial values for [VariableDataPrototypes/ParameterDataPrototypes](#) on different conceptual levels. That is, although defined for the same [VariableDataPrototype/ParameterDataPrototype](#), an initial value defined on one conceptual level can “supersede” the definition of another initial value on a different conceptual level provided that the priority of the first is higher than the priority of the latter.

The meaning of “supersede” in this context is that that the definition of an initial value on a specific conceptual level is the only relevant definition of an initial value on that level.

[TPS_SWCT_01518] Priority of initial value definition with respect to conceptual levels [Any initial value defined in the context of a conceptual level of lower priority is ignored!]()

[TPS_SWCT_01182] Conceptual levels for the definition of initial values [The following conceptual levels for the definition of initial values exist:

1. It is possible to aggregate an `initValue` directly at the definition of any `VariableDataPrototype/ParameterDataPrototype`.
2. It is possible to aggregate an `initValue` at the level of a `ComSpec`, namely:
 - `NonqueuedSenderComSpec`
 - `NonqueuedReceiverComSpec`
 - `ParameterProvideComSpec`
 - `ParameterRequireComSpec`
 - `NvRequireComSpec`
3. It is possible to aggregate a `implInitValue` and an `appInitValue` at the definition of a `CalibrationParameterValue`.

The priority of one definition of an initial value over another is reflected by the numerical order of the above enumeration, e.g. a definition on level 2 supersedes a definition on level 1. `]()`

5.7.2 Initial Value Representation

[TPS_SWCT_01183] Actual value of an `initValue` shall be interpreted according to the `AutosarDataType` `[` A `DataPrototype` can be typed by either an `ApplicationDataType` or else an `ImplementationDataType`. Therefore, the actual value of an `initValue` shall be interpreted according to the `AutosarDataType` that types the `DataPrototype`.

That is, if the `DataPrototype` is typed by an `ApplicationDataType` the value shall be interpreted as a physical value while if the `DataPrototype` is typed by an `ImplementationDataType` the value is to be interpreted as the direct numerical representation. `](RS_SWCT_03216, RS_SWCT_03217)`

[TPS_SWCT_01184] `ApplicationPrimitiveDataTypes` with category `VALUE` `[` In case of `ApplicationPrimitiveDataTypes` with category `VALUE` it is sufficient if the `initValues` are provided as physical values only because the RTE Generator should be able to evaluate the related `CompuMethod` appropriately. `](RS_SWCT_03216, RS_SWCT_03217)`

Please note that `DataPrototypes` that refer to `CompuMethods` of category `SCALE_LINEAR_AND_TEXTTABLE` (or similar) shall be initialized by means of the definition of several `ApplicationValueSpecification.swValueCont.swValues-Phys.vtf`.

Depending on the evaluation of the binding expression either a numerical value or a string is taken to initialize the `DataPrototype`.

[TPS_SWCT_01185] `initValues` for Compound Primitive Data Types `[` The definition of `initValues` in the numerical representation for Compound Primitive

Data Type (see section 5.6) is done such that the `initValues` have to be provided as a `RecordValueSpecification` respectively an `ArrayValueSpecification` or `NumericalRuleBasedValueSpecification` matching to the related `ImplementationDataType`. The additional representation can be provided and associated by means of a `ConstantSpecificationMapping`. `](RS_SWCT_03216)`

[constr_1221] `DataPrototype` is typed by an `ApplicationPrimitiveDataType` `[` If a `DataPrototype` is typed by an `ApplicationPrimitiveDataType` its `initValue` shall be provided by an `ApplicationValueSpecification`. If the underlying `ApplicationPrimitiveDataType` represents an enumeration, the value provided shall match to one of the applicable text values (`vt`, `short-Label`, `symbol`) defined by the applicable `CompuScales`. `]()`

[constr_1385] `DataPrototype` is typed by an `ImplementationDataType` `[` If a `DataPrototype` is typed by an `ImplementationDataType` its `initValue` shall **not** be provided by an `ApplicationValueSpecification`. `]()`

[constr_1222] `category` of an `AutosarDataType` used to type a `DataPrototype` is set to `STRING` `[` If the `category` of an `AutosarDataType` used to type a `DataPrototype` is set to `STRING` the `ApplicationValueSpecification` used to initialize the `DataPrototype` shall be of `category STRING`. `]()`

[constr_1223] `DataPrototype` is typed by an `ApplicationRecordDataType` `[` If a `DataPrototype` is typed by an `ApplicationRecordDataType` the corresponding `initValue` shall be provided by a `RecordValueSpecification`. `]()`

[constr_1224] `DataPrototype` is typed by an `ApplicationArrayDataType` `[` If a `DataPrototype` is typed by an `ApplicationArrayDataType` the corresponding `initValue` shall be provided by an `ArrayValueSpecification` or `ApplicationRuleBasedValueSpecification`. `]()`

5.7.3 Constant Specification Mapping

[TPS_SWCT_01186] `ConstantSpecificationMapping` `[` The `ConstantSpecificationMapping` is used to associate `ValueSpecifications` defined in the implementation domain with corresponding `ValueSpecifications` defined in the application domain.

To make this possible the `ValueSpecification` actually needs to be a `ConstantReference`. The `ConstantSpecification` referenced by the `ConstantReference` is also the target of the references owned by `ConstantSpecificationMapping`. `]()`

[constr_1029] `ConstantSpecificationMapping` and `ConstantSpecification` `[` It is required that one `ConstantSpecification` referenced from a `ConstantSpecificationMapping` needs to be defined in the application domain (`applConstant`) and the other referenced `ConstantSpecification` needs to be defined in the implementation domain (`implConstant`). `]()`

[TPS_SWCT_01187] **ConstantSpecificationMappingSet** referenced by the **InternalBehavior** [In most cases the meta-class **ConstantSpecificationMappingSet** will be referenced by the **InternalBehavior**. This **ConstantSpecificationMappingSet** contains the applicable **ConstantSpecificationMappings**.]()

However, in some specializations the software-components will not have an **InternalBehavior**:

- [constr_1030] **ParameterSwComponentType** references **ConstantSpecificationMappingSet** [**ParameterSwComponentType**: here the **ConstantSpecificationMappingSet** is directly associated by the **ParameterSwComponentType**.]()
- [constr_1031] **NvBlockSwComponentType** references **ConstantSpecificationMappingSet** [**NvBlockSwComponentType**: in this case the **ConstantSpecificationMappingSet** is associated with the aggregated **NvBlockDescriptor**.]()

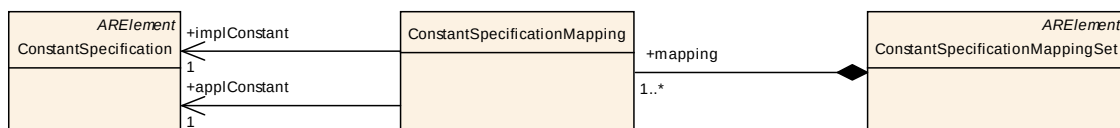


Figure 5.59: Constant Mapping

Class	ConstantSpecificationMapping			
Package	M2::AUTOSARTemplates::CommonStructure::Constants			
Note	This meta-class is used to create an association of two ConstantSpecifications. One ConstantSpecification is supposed to be defined in the application domain while the other should be defined in the implementation domain. Hence the ConstantSpecificationMapping needs to be used where a ConstantSpecification defined in one domain needs to be associated to a ConstantSpecification in the other domain. This information is crucial for the RTE generator.			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
applConstant	ConstantSpecification	1	ref	A ConstantSpecification defined in the application domain.
implConstant	ConstantSpecification	1	ref	A ConstantSpecification defined in the implementation domain.

Table 5.131: ConstantSpecificationMapping

Class	ConstantSpecificationMappingSet			
Package	M2::AUTOSARTemplates::CommonStructure::Constants			
Note	This meta-class represents the ability to map two ConstantSpecifications to each others. One ConstantSpecification is supposed to be described in the application domain and the other should be described in the implementation domain. Tags: atp.recommendedPackage=ConstantSpecificationMappingSets			
Base	ARElement , ARObject , CollectableElement , Identifiable , Multilanguage Referrable , PackageableElement , Referrable			
Attribute	Datatype	Mul.	Kind	Note
mapping	ConstantSpecificationMapping	1..*	aggr	ConstantSpecificationMappings owned by the ConstantSpecificationMappingSet.

Table 5.132: ConstantSpecificationMappingSet

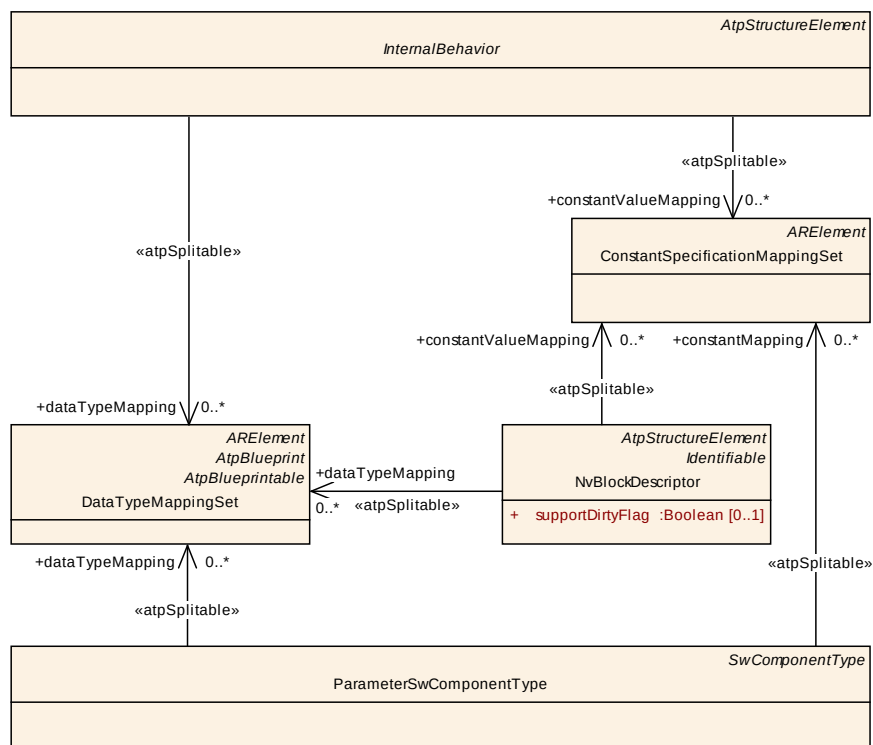


Figure 5.60: Aggregation of [ConstantSpecificationMappingSet](#)

5.7.4 Initial Values For CalibrationParameters

[TPS_SWCT_01188] Definition of calibration data sets through RTE-generator and compiler [It is possible to provide sets of initial values for calibration parameters which are instance specific, thus overriding any initial values predefined by a [ParameterDataPrototype](#), [ParameterRequireComSpec](#) or a [ParameterProvideComSpec](#).

This allows to create the calibration data sets through RTE-generator and compiler. These initial values are specified in [CalibrationParameterValueSet](#) and [CalibrationParameterValue](#). The latter aggregates a [ValueSpecification](#) in two different roles:

- [applInitValue](#) for data structured according to [ApplicationDataType](#). In this case the values are defined in the physical domain.
- [implInitValue](#) for data structured according to [ImplementationDataType](#). In this case the values are defined in the numerical domain.

]([RS_SWCT_03175](#))

Anyhow, these initial values can be imported from e.g. an ASAM CDF file.

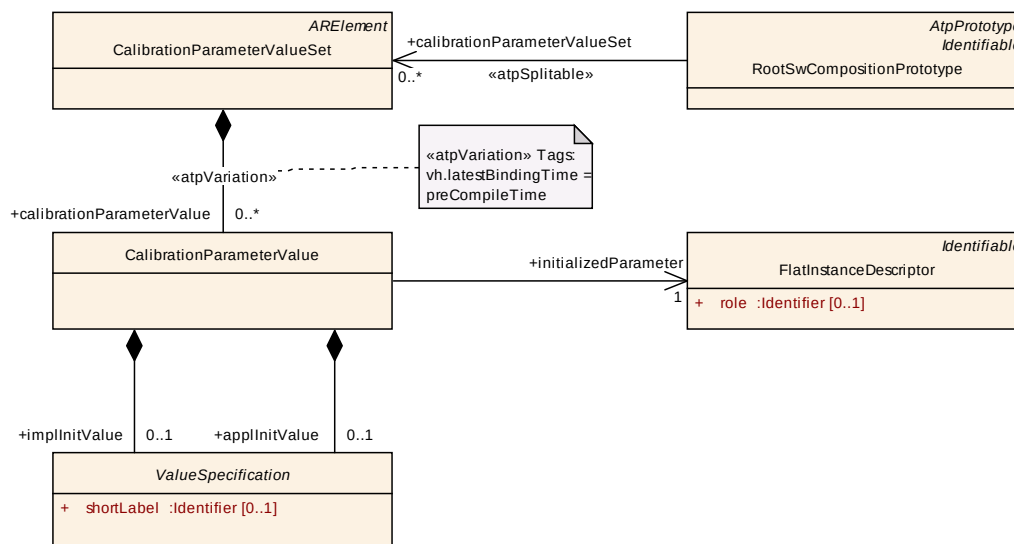


Figure 5.61: Calibration Parameter Values

Class	CalibrationParameterValueSet			
Package	M2::AUTOSARTemplates::SWComponentTemplate::MeasurementAndCalibration::CalibrationParameterValues			
Note	Specification of a constant that can be part of a package, i.e. it can be defined stand-alone. Tags: atp.recommendedPackage=CalibrationParameterValueSets			
Base	ARElement , ARObject , CollectableElement , Identifiable , Multilanguage , Referrable , PackageableElement , Referrable			
Attribute	Datatype	Mul.	Kind	Note
calibration Parameter Value	CalibrationPara meterValue	*	aggr	This represents single CalibrationParameterValues in the CalibrationParameterValueSet. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime

Table 5.133: CalibrationParameterValueSet

Class	CalibrationParameterValue			
Package	M2::AUTOSARTemplates::SWComponentTemplate::MeasurementAndCalibration::CalibrationParameterValues			
Note	Specifies instance specific calibration parameter values used to initialize the memory objects implementing calibration parameters in the generated RTE code. RTE generator will use the implInitValue to override the initial values specified for the DataPrototypes of a component type. The applInitValue is used to exchange init values with the component vendor not publishing the transformation algorithm between ApplicationDataTypes and ImplementationDataTypes or defining a instance specific initialization of components which are only defined with ApplicationDataTypes. Note: If both representations of init values are available these need to represent the same content. Note further that in this case an explicit mapping of ValueSpecification is not implemented because calibration parameters are delivered back after the calibration phase.			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
applInitValue	ValueSpecification	0..1	aggr	This is the initial value specification structured according to the ApplicationDataType
implInitValue	ValueSpecification	0..1	aggr	This is the initial value specification structured according to the ImplementationDataType
initializedParameter	FlatInstanceDescriptor	1	ref	This represents the parameter that is initialized by the CalibrationParameterValue.

Table 5.134: CalibrationParameterValue

6 Compatibility

6.1 Introduction

In order to connect `PortPrototypes` of `SwComponentTypes`, the compatibility of `PortPrototypes` needs to be verified. This section defines the basic rules for formal compatibility of `PortPrototypes`.

Compatibility will be defined bottom-up, i.e. first the rules for compatible `Autosar-DataTypes` are set up, then the rules for the different types of `PortInterfaces` are derived.

Another aspect of compatibility is the question whether two model-elements (e.g. `ApplicationDataType` vs. `ImplementationDataType`) can be mapped to each other.

For the compatibility of `PortInterfaces` basically two options apply:

1. finding of matching pairs of elements of `PortInterfaces` is based on matching `shortName` plus the application of compatibility rules for their attributes.
2. a `PortInterfaceMapping` can be taken to declare two elements of `PortPrototypes` as compatible without applying further formal checks.

6.2 Compatibility of Data Types

The AUTOSAR meta model defines a number of meta-classes (e.g. `ApplicationPrimitiveDataType`) that eventually refer to a set of attributes (e.g. a lower boundary for its values) relevant for compatibility checking.

Instantiating a data-type related meta-class defines a data type on M1 level (e.g. `temperatureType`). In other words: `ApplicationPrimitiveDataType` is an M2 artifact; it is taken as the template for creating a corresponding M1 artifact `temperatureType`.

In this context, the issue of compatibility refers to the M1 objects, i.e. the instances of sub-classes of `AutosarDataType` need to be considered. For this purpose the relevant part of the AUTOSAR meta-model need to be fully explored with respect to compatibility.

6.2.1 ApplicationDataType

6.2.1.1 ApplicationPrimitiveDataType

[constr_1047] Compatibility of `ApplicationPrimitiveDataTypes` [Instances of `ApplicationPrimitiveDataType` are compatible if and only if one of the following conditions applies:

1. All of the following subconditions apply:
 - (a) They have the same `category` (see table in figure 5.8).
 - (b) The `swDataDefProps` attached to the M1 data types are compatible. The meaning of this statement is explained in section 6.2.4.
2. In the context of using the `ApplicationPrimitiveDataType`, a `DataPrototypeMapping` exists that refers to a `DataPrototype` typed by one of the `ApplicationPrimitiveDataTypes` in the role `firstDataPrototype` and to another `DataPrototype` typed by the other `ApplicationPrimitiveDataType` in the role `secondDataPrototype`.
3. In the context of using the `ApplicationPrimitiveDataType`, a `DataPrototypeMapping` exists that refers to a `DataPrototype` typed by the `ApplicationPrimitiveDataType` in the role `secondDataPrototype` and to another `DataPrototype` typed by an `ApplicationCompositeDataType` in the role `firstDataPrototype` and additionally for the side of the `ApplicationCompositeDataType` a corresponding `ApplicationCompositeDataTypeSubElementRef` exists in the role `firstElement` that in turn references an `ApplicationCompositeElementDataPrototype`.

]()

Please note that it is **not** required that the `shortNames` of two data types shall be identical in order to consider the two data types as compatible.

6.2.1.2 ApplicationCompositeDataType

An instance of an `ApplicationRecordDataType` is never compatible to an instance of an `ApplicationArrayDataType` **unless** a `PortInterfaceMapping` exists that details the terms of compatibility (see [TPS_SWCT_01543]).

[constr_1048] Compatibility of ApplicationRecordDataTypes [Instances of `ApplicationRecordDataTypes` are compatible if and only if one of the following conditions applies:

1. All *elements at the same record position* are of compatible `AutosarDataTypes` either `ApplicationCompositeDataTypes` or `ApplicationPrimitiveDataTypes`).
2. In the context of a `DataPrototypeMapping`, for each `ApplicationRecordElement` of the required `ApplicationRecordDataType` a `SubElementMapping` exists such that a `ApplicationCompositeDataTypeSubElementRef` in the role `firstElement` or `secondElement` exists that references the required `ApplicationRecordElement` **and** a corresponding `ApplicationCompositeDataTypeSubElementRef` exists in the **other** role (i.e. `secondElement` or `firstElement`) that in turn references an `ApplicationRecordElement` of the provided `ApplicationRecordDataType`.

]()

[constr_1049] Compatibility of `ApplicationArrayDataTypes` [Instances of `ApplicationArrayDataType` are compatible if and only if one of the following conditions applies:

1. All of the following subconditions apply:
 - (a) Their `elements` are of a compatible `AutosarDataTypes` (either `ApplicationCompositeDataTypes` or `ApplicationPrimitiveDataTypes`).
 - (b) The attributes `maxNumberOfElements` and `arraySizeSemantics` (given the existence) have identical values.
2. In the context of a `DataPrototypeMapping`, for the `ApplicationArrayElement` of the required `ApplicationArrayDataType` a `SubElementMapping` exists such that a `ApplicationCompositeDataTypeSubElementRef` in the role `firstElement` or `secondElement` exists that references the required `ApplicationArrayElement` **and** a corresponding `ApplicationCompositeDataTypeSubElementRef` exists in the **other** role (i.e. `secondElement` or `firstElement`) that in turn references an `ApplicationArrayElement` of the provided `ApplicationArrayDataType`.

]()

6.2.2 ImplementationDataType

[constr_1050] Compatibility of `ImplementationDataTypes` [Instances of `ImplementationDataType` are compatible if and only if after all type-references are resolved one of the following rules apply:

1. All of the following subconditions apply:
 - (a) They have the same `category` (see table 5.18)
 - (b) They have the identical structure (this refers to `ImplementationDataElement` and their `subElements`).
 - (c) The attributes `arraySize` and `arraySizeSemantics` have (given the existence) identical values.
 - (d) The `swDataDefProps` attached to the M1 data types are compatible. The meaning of this statement is explained in section 6.2.4.
2. In the context of using the `ImplementationDataType`, a `DataPrototypeMapping` exists that refers to a `DataPrototype` typed by one of the `ImplementationDataTypes` in the role `firstDataPrototype` and to another `DataPrototype` typed by the other `ImplementationDataType` in the role `secondDataPrototype`.

3. In the context of using the `ImplementationDataType`, a `DataPrototypeMapping` exists that refers to a `DataPrototype` typed by the `ImplementationDataTypes` in the role `secondDataPrototype` and to another `DataPrototype` typed by an `ImplementationDataType` with a `subElement` in the role `firstDataPrototype` and additionally for the side of the `ImplementationDataType` with a `subElement` a corresponding `ImplementationDataTypeSubElementRef` exists in the role `firstElement` that in turn references an `ImplementationDataTypeElement`.

⌋()

Please note that it is **not** required that the `shortNames` of two data types shall be identical in order to consider the two data types as compatible.

The following constraint applies for the case that mode manager and mode user are using different `ImplementationDataTypes`. From the point of view of the RTE there is only the necessity that all possible numbers used to represent `ModeDeclarations` of the mode manager has to fit into the range of the data type used for the mode user.

[constr_1168] Compatibility of `ImplementationDataTypes` used used in the `ModeRequestTypeMap` ⌈ Both `ImplementationDataTypes` shall fulfill [constr_1167].

In addition to that, the possible numbers used for representing `ModeDeclarations` on the side of the mode manager shall match the supported range of the `ImplementationDataType` used for representing `ModeDeclarations` on the side of the mode user (see [constr_1075]). ⌋()

6.2.3 Compatibility of `SwBaseType`

[constr_1220] Compatibility of `SwBaseType` ⌈ Two `SwBaseTypes` are compatible if and only if attributes `baseTypeSize` respectively `maxBaseTypeSize`, `byteOrder`, `memAlignment`, `baseTypeEncoding`, and `nativeDeclaration` have identical values. ⌋()

6.2.4 Compatibility of `SwDataDefProps`

[constr_1051] Compatibility of `SwDataDefProps` ⌈ `SwDataDefProps` are compatible if and only if:

1. They refer to compatible `Unit` definitions, or neither of them has an associated `Unit`.
2. They refer to compatible conversion methods (see chapter 6.2.4.5) or neither of them associates such a method.

3. One of the following conditions apply to `ValueSpecifications` aggregated in the role `invalidValue` for being considered compatible (after following and resolving indirections created by `ConstantReference`):
 - (a) both are `ApplicationValueSpecifications` and the values are compatible according to [TPS_GST_02501].
 - (b) both are `NumericalValueSpecifications` and the values are compatible according to [TPS_GST_02501].
 - (c) both are `TextValueSpecifications` and the values are identical.
 - (d) both are `ArrayValueSpecifications` and the values are identical.
 - (e) both are `RecordValueSpecifications` and the values are identical.
 - (f) if one is a `NumericalValueSpecification` and the other one is an `ApplicationValueSpecification` then the check for compatibility shall apply the `CompuMethod` on the physical value such that a comparison on the implementation level becomes possible. [TPS_GST_02501] applies¹.
4. They refer to compatible data constraints `dataConstr`.
5. They refer to compatible `swRecordLayouts`

All other attributes (e.g. `swCalibrationAccess` do not affect compatibility). `]()`

6.2.4.1 Compatibility of Units

[constr_1052] Compatibility of Units [Two `Unit` definitions are compatible if and only if:

1. They have compatible (see [TPS_GST_02501]) values of attributes `factorSiToUnit` and `offsetSiToUnit`.
2. They either refer to identical definitions of `PhysicalDimension` or neither of them associates a `PhysicalDimension`.

`]()`

Please note that it is **not** required that the `shortNames` of two `Units` shall be identical in order to consider the two units as compatible.

[TPS_SWCT_01492] Default values for `factorSiToUnit` and `offsetSiToUnit`
[The default value of attribute `Unit.factorSiToUnit` is 1.
The default value of attribute `Unit.offsetSiToUnit` is 0. `]()`

Further constraints apply specifically for the handling of `Units` in the context of assigning a `ValueSpecification` to a given `AutosarDataPrototype`:

¹if one is a `NumericalValueSpecification` and the other one is an `ApplicationValueSpecification` and the application of the `CompuMethod` on the side of the `ApplicationValueSpecification` does not yield a valid number a comparison is not possible.

[constr_1391] Compatibility of Units in the context of assignment using an `ApplicationValueSpecification` [If an `ApplicationValueSpecification` is used in the context of an assignment to an `AutosarDataPrototype` then the `ApplicationValueSpecification.swValueCont.unit` shall be compatible to the `Unit` used in the definition of the given `AutosarDataPrototype`, i.e. `AutosarDataType.swDataDefProps.unit`.]()

[constr_1392] Compatibility of Units in the context of assignment using an `ApplicationRuleBasedValueSpecification` [If an `ApplicationRuleBasedValueSpecification` is used in the context of an assignment to an `AutosarDataPrototype` then the `ApplicationRuleBasedValueSpecification.swValueCont.unit` shall be compatible to the `Unit` used in the definition of the given `AutosarDataPrototype`, i.e. `AutosarDataType.swDataDefProps.unit`.]()

[constr_1393] Existence of `RuleBasedValueCont.unit` [For every `RuleBasedValueCont` the attribute `unit` shall exist.]()

Please note that the multiplicity of `RuleBasedValueCont.unit` is set to 0..1 while the multiplicity of the corresponding `SwValueCont.unit` is set to 1.

This inconsistency cannot be resolved by increasing the lower multiplicity of `RuleBasedValueCont.unit` because this would create an incompatible XML Schema. However, the creation of [constr_1393] effectively yields the same result.

6.2.4.2 Compatibility of PhysicalDimensions

[constr_1053] Compatibility of `PhysicalDimensions` [Two `PhysicalDimension` definitions are compatible if and only if the values of

- `lengthExp`
- `massExp`
- `timeExp`
- `currentExp`
- `temperatureExp`
- `molarAmountExp`
- `luminousIntensityExp`

are identical and **either** the `shortNames` are identical **or** a `PhysicalDimensionMapping` exists that maps one of the `PhysicalDimensions` in the role `firstPhysicalDimension` and the other `PhysicalDimension` in the role `secondPhysicalDimension`.]()

For clarification, there are some physical dimensions around that share the identical values for the exponents but still have a completely different meaning and shall therefore not be considered compatible. For precisely this reason [constr_1053] **requires**

the `shortNames` of two `PhysicalDimensions` to be identical as a prerequisite for compatibility.

For example, there are at least two physical dimensions that share the values of

- `lengthExp = 2`
- `massExp = 1`
- `timeExp = -2`
- `currentExp = 0`
- `temperatureExp = 0`
- `molarAmountExp = 0`
- `luminousIntensityExp = 0`

The unit described by this set of exponents is usually referred to as “Nm” for *newton-meter* and it can be used for *torque* just as well as for *energy*. Obviously, two `Units` shall never be considered compatible if one refers to *torque* and the other one refers to *energy*.

6.2.4.3 Compatibility of Data Constraints

The compatibility of two `DataConstrs` depends on the context in which the owning data elements are connected:

[constr_1126] Compatibility of `DataConstrs` [The `DataConstr` (e.g. the limits) defined by the type of the providing data element shall be within the constraints defined by the type of the requiring data element.]()

In addition, it is always allowed if the requiring element defines no constraints.

[constr_1278] `PhysConstrs` references a `Unit` [`DataConstrs` are only compatible if the `DataConstr.dataConstrRule.physConstrs.unit` are compatible or neither `DataConstr.dataConstrRule.physConstrs.unit` exist.]()

[constr_1054] No `DataConstr` available at the provider [If the provider defines no constraints it is only compatible with a receiver which also defines no constraints at all.]()

In other words, this is not a compatibility rule for the types but for the data prototypes.

6.2.4.4 Compatibility in case of `ImplementationDataType`

If the `SwDataDefProps` are owned by an `ImplementationDataType` further conditions shall be met to ensure compatibility.

Note that depending on the `category` of the `ImplementationDataType`, at most one of these four constraints is actually relevant:

1. `category` **[constr_1055] ImplementationDataType has category VALUE** [The attributes `baseType` shall refer to a compatible `SwBaseType`]() (see explanation in the following rule). The rules regarding the compatibility of `SwBaseTypes` are covered by [constr_1220].
2. `category` **TYPE_REFERENCE: [constr_1056] ImplementationDataType has category TYPE_REFERENCE** [The `ImplementationDataTypes` referenced by the attributes `SwDataDefProps.implementationDataType` shall be compatible .]()
3. `category` **DATA_REFERENCE: [constr_1057] ImplementationDataType has category DATA_REFERENCE** [The attributes `SwDataDefProps.swPointerTargetProps` shall have identical `targetCategory` and shall refer to `SwDataDefProps` where all attributes are identical]() (in other words, the target types of the pointers shall be identical, not only compatible).
4. `category` **FUNCTION_REFERENCE: [constr_1058] Implementation-DataType has category FUNCTION_REFERENCE** [The attributes `SwDataDefProps.swPointerTargetProps.functionPointerSignature` shall refer to `BswModuleEntry`s which each resolve to the **same function signature**.]()

Please note that the term “same signature” refers to the following predicates:

- same number of arguments
- return values and arguments shall have **identical** - not only *compatible* - data types

Two `SwBaseTypes` are compatible (in the sense of allowing a connection of ports via the RTE) if a simple conversion rule exists between the two types in the underlying programming language.

Admittedly, this is a rather weak condition. But because the definition of `SwBaseTypes` can contain a `nativeDeclaration` it is not possible to state this rule more specifically.

However, conversion between base types is considered as a less common use case than the simple case that the connected types just contain two identical `SwBaseTypes` (which is of course included in the rule).

Please note, that in addition the existence of `ApplicationDataTypes` also constraints the possible `SwBaseTypes` via the compatibility rules for the mapping between `ApplicationDataTypes` and `ImplementationDataType` as will be explained in more detail in chapter 6.2.5.

6.2.4.5 Compatibility of CompuMethods

[constr_1163] Compatibility of CompuMethods [Two `CompuMethod` definitions are compatible if and only if all attributes **except**

- `shortName`
- `desc`
- `introduction`
- `longName`
- `adminData`
- `annotation`
- `displayFormat`

are **identical and** the `compuScales` and `units` are compatible.]()

[constr_1153] Applicability of compatibility requirements for CompuScales [Compatibility requirements for `CompuScales` shall only apply for `CompuScales` where the `category` of the enclosing `CompuMethod` is one of the following:

- `SCALE_LINEAR_AND_TEXTTABLE`
- `SCALE_RATIONAL_AND_TEXTTABLE`
- `TEXTTABLE`
- `TAB_NOINTP`
- `BITFIELD_TEXTTABLE`
- `LINEAR`
- `RAT_FUNC`
- `IDENTICAL`

]()

[constr_1154] Compatibility of CompuScales for sender-receiver communication and similar use cases [For sender-receiver communication and similar use cases, it is required that the set of `CompuScales` defined in the `CompuMethod` of the provider of the communication (i.e. on the side of the `PPortPrototype`) shall be a subset of the set of `CompuScales` defined in the `CompuMethod` on the required side (i.e. on the side of the `RPortPrototype`).]()

[constr_1155] Compatibility of CompuScales for client-server communication [For client-server communication, the following rules apply:

For `arguments` of direction `IN` the `CompuScales` defined in the `CompuMethod` of the client (i.e. on the side of the `RPortPrototype`) shall be a subset of the set of

`CompuScales` defined in the `CompuMethod` supported at the server (i.e. on the side of the `PPortPrototype`).

For `arguments` of the direction `OUT` the set of `CompuScales` defined in the `CompuMethod` of the server (i.e. on the side of the `PPortPrototype`) shall be a subset of the set of `CompuScales` defined in the `CompuMethod` supported at the client (i.e. on the side of the `RPortPrototype`).

For `arguments` of direction `INOUT` the set of `CompuScales` defined in the `CompuMethod` of server and client shall be identical. $\downarrow()$

[constr_1156] Relevance of “names” of `CompuScales` $\lceil$ `CompuScales` which contribute to tabular conversion by having a `compuConst` are compatible **if and only if** the “names” of the `compuScales`, (namely `shortLabel`, `compuConst` and `symbol`) are equal. If the scale has no `compuConst`, “names” of `CompuScales` are not relevant for compatibility. $\downarrow()$

[constr_1157] Applicability of constraints of `CompuScales` $\lceil$ The constraints **[constr_1154]**, **[constr_1155]**, and **[constr_1156]** shall **only** apply in the absence of a `TextTableMapping` which shall take precedence regarding the compatibility if it exists. $\downarrow()$

[constr_1176] Compatibility of `CompuScales` of category `LINEAR` and `RAT_FUNC` $\lceil$ `CompuScales` of category `LINEAR` and `RAT_FUNC` are considered compatible if they yield the same conversion. $\downarrow()$

In other words, $\frac{n_0+n_1*phys}{d_0+d_1*phys}$ is compatible to $\frac{N_0+N_1*phys}{D_0}$ if $n_0 \sim N_0$ && $n_1 \sim N_1$ && $d_0 \sim D_0$ && $d_1 \sim 0$.

Note that $\sim$ indicates compatibility of numerical values according to [TPS_GST_02501].

[constr_1192] Compatibility of “`IDENTICAL`” to “`RAT_FUNC`” or “`LINEAR`” $\lceil$ Similar to **[constr_1176]**, a `CompuScale` where the `category` of the enclosing `CompuMethod` is set to `IDENTICAL` is considered compatible to a `CompuScale` where the `category` of the enclosing `CompuMethod` is set to `RAT_FUNC` or `LINEAR` if the following rule applies:

$$int = \frac{N_0+N_1*phys+N_i*phys^i}{D_0+D_1*phys+D_i*phys^i} = phys$$

$\downarrow()$

This is the case for

$$N_0 \sim 0 \text{ \&\& } D_0 \sim 1 \text{ \&\& } N_1 \sim 1 \text{ \&\& } D_1 \sim 0 \text{ \&\& } N_i \sim D_i \sim 0 \forall i > 1.$$

6.2.4.6 Compatibility of Record Layouts

[constr_1162] Compatibility of `SwRecordLayouts` $\lceil$ Two `SwRecordLayout` definitions are compatible if and only if all attributes **except**

- `shortName`

- desc
- introduction
- longName
- adminData
- annotation

are **identical**. $\downarrow()$

6.2.5 Compatibility of `ApplicationDataType` and `ImplementationDataType`

The usage of `ApplicationDataTypes` implies that also a corresponding `ImplementationDataType` exists at a certain point in time. The `ImplementationDataType` is required as the basis for configuring and generating the RTE and/or contract phase header files.

[TPS_SWCT_01461] Existence of `ImplementationDataType` [The existence of `ImplementationDataTypes` is **not** required until the methodology step of generating an RTE or executing the RTE contract phase. Before arriving at this step in the methodology, it is perfectly feasible to use only `ApplicationDataTypes` for describing the semantics of software-components. $\downarrow()$

As a consequence, it is necessary to define compatibility rules that unambiguously clarify the conformance of an `ApplicationDataType` with an `ImplementationDataType` and vice versa.

Please note that this kind of compatibility also supports situations where e.g. a `dataElement` typed by an `ApplicationDataType` without a corresponding `ImplementationDataType` in a `PPortPrototype` should be connected to a `dataElement` typed by an `ImplementationDataType` in an `RPortPrototype`.

In general, the compatibility rules for allowing a data type mapping are the same as the rules for connections. Exceptions are explicitly stated in the rules below.

Several rules depend on the `category` of the data types:

1. As a general rule, if an `ImplementationDataType` of `category TYPE_REFERENCE` is targeted by a type mapping or port connection all the rules given below apply to the `ImplementationDataType` which is finally valid after resolving all such references.

This is not repeated in all rules. As an example, if we say that something can be mapped/connected to an `ImplementationDataType` of `category VALUE` this shall include the possibility of mapping/connecting to an `ImplementationDataType` of `category TYPE_REFERENCE` which refers to another `ImplementationDataType` of `category VALUE`.

2. **[constr_1059] Compatibility of data types with category VALUE** [An `ApplicationDataType` of category VALUE can only be mapped/connected to an `ImplementationDataType` which also has category VALUE.]()

In this case, the `ImplementationDataType.baseType` shall be able to express all the numerical values required by the `ApplicationDataType`, see Figure 5.6.

This condition is fulfilled if the numerical range which can be expressed by the `SwBaseType` at least covers the range defined by the limits in `ApplicationDataType.swDataDefProps.dataConstr` (which are either internal limits or physical limits to be converted via the `CompuMethod` which also has to be provided by the `ApplicationDataType`).

The condition is also fulfilled if the `SwBaseType` covers the range defined in the `CompuMethod` for an enumeration (see 5.5.1.3).

Note that for sender-receiver communication of a data element via a network there is the possibility to reduce the numerical range against what has been defined via the corresponding data type. However, this is not achieved via mapping to another `ImplementationDataType` at the data element itself but via the `networkRepresentation` of the `ComSpec` (for further explanation of this aspect see section 4.5.1).

3. **[constr_1060] Compatibility of data types with category ARRAY, VAL_BLK** [An `ApplicationDataType` of category ARRAY, VAL_BLK can only be mapped/connected to

- an `ImplementationDataType` of category ARRAY or
- an `ImplementationDataType` that represents a Variable-Size Array Data Type (see [TPS_SWCT_01610]).

]()

In this case, the array size, the `arraySizeSemantics` (given that it exists) and the type of the array elements of the `ImplementationDataType` shall be such that they can be mapped resp. transferred 1:1 by order to the corresponding application data and vice versa.

Note that in case of mapping between arrays it is not required that a `DataTypeMap` exists between the data types of the array elements or that the respective `ShortNames` are identical.

4. **[constr_1061] Compatibility of data types with category STRUCTURE** [An `ApplicationDataType` of category STRUCTURE can only be mapped/connected to an `ImplementationDataType` of category STRUCTURE.]()

This means, that the corresponding pairs of elements shall also have compatible types. Note that it is not required that the data types of the single elements have identical `ShortNames` or that a `DataTypeMap` exists for each pair of single element.

5. **[constr_1063] Compatibility of data types with category BOOLEAN** [An `ApplicationDataType` of category `BOOLEAN` can only be mapped/connected to an `ImplementationDataType` of category `VALUE`.]()
6. **[constr_1064] Compatibility of data types with category COM_AXIS, RES_AXIS, CURVE, MAP, CUBOID, CUBE_4, or CUBE_5** [An `ApplicationDataType` of category `COM_AXIS`, `RES_AXIS`, `CURVE`, `MAP`, `CUBOID`, `CUBE_4`, or `CUBE_5` can only be mapped/connected to an `ImplementationDataType` of category `STRUCTURE` or `ARRAY`.]()

There are several possibilities how to express these types via plain or nested arrays and/or structures on implementation level.

Some examples are given in 5.4.4. In any case, the primitive elements of the implementation type shall fit (by their order in memory) to the corresponding `RecordLayout`.

It is not required, to define `DataTypeMaps` for the sub-elements or both representations.

7. **[constr_1066] Forbidden mappings to ImplementationDataType** [An `ApplicationDataType` shall never be mapped to an `ImplementationDataType` of of category `UNION`, `DATA_REFERENCE`, or `FUNCTION_REFERENCE`.]()

Concerning the `SwDataDefProps` of an `ApplicationDataType` instance resp. an `ImplementationDataType` instance which shall be mapped/connected on M1, we refer to the table shown in figure 5.39. The following rules apply:

1. The cases where the `ImplementationDataType` is not allowed to set a property but only “inherits” it from the `ApplicationDataType` are not relevant for compatibility. These attributes are simply not allowed in the `ImplementationDataType`.
2. In case that only the `ImplementationDataType` may “define” the property this definition shall fit into the semantical requirements given by the `ApplicationDataType` in order to make the two types compatible.

This is namely important for the attribute `baseType` and is explained above in the rule for types of category `VALUE`.

3. In case the `ImplementationDataType` may “add” a property it may only add but not change a property defined by the `ApplicationDataType` (namely `note`, `displayFormat`, and `swImplPolicy`) in order to be compatible.

This means that the respective computation methods can be defined in only one of the types in order to be compatible. In all other cases, only the `ApplicationDataType` may define the computation method.

4. For the compatibility with respect to connectors there are some additional rules for the values of the attribute `swImplPolicy` which are considered general rules on the level of `DataPrototypes` and `PortInterfaces`.

Therefore these additional rules are explained in chapter 6.3 and chapter 6.4.4.

5. The case that an `ImplementationDataType` may “redefine” a property which is already set by the `ApplicationDataType` is not considered as relevant for the compatibility with respect to mapping of the types in general but of course there may be project specific rules as to which redefinition is allowed (e.g. for `swAddrMethod` or `dataConstr`). See also 5.5.3 about data constraints.
6. For the compatibility with respect to connectors the attribute `dataConstr` shall be treated in the same way as for compatibility of data types in general, for more details please refer to 6.2.4.

6.3 Compatibility of Variable Data Prototypes and Parameter Data Prototypes

[constr_1068] Compatibility of `VariableDataPrototypes` or `ParameterDataPrototypes` typed by primitive data types [Two `VariableDataPrototypes` or `ParameterDataPrototypes` of `ApplicationPrimitiveDataTypes` or `ImplementationDataTypes` of category `VALUE`, `BOOLEAN`, or `STRING` are compatible if and only if one of the following conditions applies:

1. All of the following subconditions apply:
 - (a) They are typed by (read “refer to”) compatible `AutosarDataTypes`
 - (b) The two `VariableDataPrototypes` or `ParameterDataPrototypes` have identical `shortNames` This is required to map `VariableDataPrototypes` in unordered `SenderReceiverInterfaces`, `NvDataInterfaces` and `ParameterInterfaces`.
 - (c) The attribute `swImplPolicy` is either set to `queued` for both or none of the `VariableDataPrototypes`.
2. In the context of a `DataPrototypeMapping`, one of the applicable `VariableDataPrototypes` or `ParameterDataPrototypes` is referenced by the `DataPrototypeMapping` in the role `firstDataPrototype` and the other `VariableDataPrototypes` or `ParameterDataPrototypes` is referenced by the same `DataPrototypeMapping` in the role `secondDataPrototype`.

]()

[constr_1187] Compatibility of `VariableDataPrototypes` or `ParameterDataPrototypes` typed by composite data types [

`DataPrototypes` of `ApplicationCompositeDataTypes` or `ImplementationDataTypes` of category `STRUCTURE` or `ARRAY` are compatible if one of the following conditions evaluates to true:

1. The underlying `ApplicationCompositeDataTypes` or `ImplementationDataTypes` of category `STRUCTURE` or `ARRAY` are identical
2. The underlying `ApplicationCompositeDataTypes` or `ImplementationDataTypes` of category `STRUCTURE` or `ARRAY` fulfill the following condition:
 - They consist of the same number of elements **and**
 - They are composed of compatible `AutosarDataTypes` (either `ApplicationCompositeDataTypes` or `ImplementationDataTypes` of category `STRUCTURE` or `ARRAY` **OR** `ApplicationPrimitiveDataTypes` or `ImplementationDataTypes` of category `VALUE`, `BOOLEAN`, or `STRING`) *in the same order* **and**
 - All attributes match exactly, with the exception of the `shortName` of the M1 `AutosarDataType`.
3. In the context of a `DataPrototypeMapping`, for each `ApplicationCompositeElementDataPrototype` of the required `DataPrototype` a `SubElementMapping` exists such that a `ApplicationCompositeDataTypeSubElementRef` in the role `firstElement` or `secondElement` exists that references the required `ApplicationCompositeElementDataPrototype` **and** a corresponding `ApplicationCompositeDataTypeSubElementRef` exists in the **other** role (i.e. `secondElement` or `firstElement`) that in turn references an `ApplicationCompositeElementDataPrototype` of the provided `ApplicationCompositeDataType`.
4. If and only if the `DataPrototype` is **not** typed by an `ApplicationDataType` but by an `ImplementationDataType`: in the context of a `DataPrototypeMapping`, for each `ImplementationDataTypeElement` of the required `DataPrototype` a `SubElementMapping` exists such that a `ImplementationDataTypeSubElementRef` in the role `firstElement` or `secondElement` exists that references the required `ImplementationDataTypeElement` **and** a corresponding `ImplementationDataTypeSubElementRef` exists in the **other** role (i.e. `secondElement` or `firstElement`) that in turn references an `ImplementationDataTypeElement` of the provided `ImplementationDataType`.

]()

6.4 Compatibility of Sender Receiver Interfaces, Parameter Interfaces and Non Volatile Data Interfaces

Please note that this compatibility requirement only satisfies static correctness which means that logical consistency is not assured (e.g. that a receiver shall process a certain data value to correctly interpret the following values).

6.4.1 Connection of Required and Provided Port via AssemblySwConnector

The compatibility of `SenderReceiverInterfaces`, `NvDataInterfaces` and `ParameterInterfaces` are considered for connecting of `PortPrototypes` with an `AssemblySwConnector`.

[constr_1069] Compatibility of `PortPrototypes` of different `DataInterfaces` in the context of `AssemblySwConnectors` [`PortPrototypes` of different `DataInterfaces` are compatible if and only if

1. One of the following conditions applies:

- (a) For each `VariableDataPrototype` or `ParameterDataPrototype` defined in the context of the `DataInterface` of the required `PortPrototype` a compatible (see [constr_1068]) `VariableDataPrototype` or `ParameterDataPrototype` exists in the `DataInterface` of the provided `PortPrototype`.

The `shortNames` of `VariableDataPrototypes` and `ParameterDataPrototypes` are used to identify the pair.

- (b) A `VariableAndParameterInterfaceMapping.dataMapping` exists for which the following conditions apply:
 - i. It is referenced by the corresponding `SwConnector`.
 - ii. It references one of the two `VariableDataPrototypes` or `ParameterDataPrototypes` in the role `firstDataPrototype` and the other in the role `secondDataPrototype`.

2. For each such pair, the values of their `isService` attributes are identical.

]()

The table 6.1 defines which `PortInterface` elements are compatible depending on the `PortInterface` type and the `swImplPolicy` attributes of the `PortInterface` elements.

6.4.2 Connection of Inner and Outer Port via DelegationSwConnector

The compatibility of `SenderReceiverInterfaces`, `NvDataInterfaces` and `ParameterInterfaces` is considered for connecting of `PortPrototypes` with a `DelegationSwConnector`.

[constr_1070] Compatibility of `PortPrototypes` of different `DataInterfaces` in the context of `DelegationSwConnectors` [`PortPrototypes` of different `DataInterfaces` are compatible if and only if

1. One of the following conditions applies:

- (a) For each `VariableDataPrototype` or `ParameterDataPrototype` defined in the context of the `DataInterface` of the required inner `PortPrototype` a compatible `VariableDataPrototype` or `ParameterDataPrototype` exists in the `DataInterface` of the required outer `PortPrototype`.

The `shortName` of `VariableDataPrototypes` and `ParameterDataPrototypes` are used to identify the pair.

[[constr_1071](#)] defines which `PortInterface` elements are compatible depending on the `PortInterface` type and the `swImplPolicy` attributes of the `PortInterface` elements.

- (b) A `VariableAndParameterInterfaceMapping.dataMapping` exists for which the following conditions apply:
 - i. It is referenced by the corresponding `SwConnector`.
 - ii. It references one of the two `VariableDataPrototypes` or `ParameterDataPrototypes` in the role `firstDataPrototype` and the other in the role `secondDataPrototype`.

2. One of the following conditions applies:

- (a) For at least one `VariableDataPrototype` or `ParameterDataPrototype` defined in the context of the `SenderReceiverInterface`, `NvDataInterface` or `ParameterInterface` of the provided inner `PortPrototype` a compatible `VariableDataPrototype` or `ParameterDataPrototype` exists in the `SenderReceiverInterface`, `NvDataInterface` or `ParameterInterface` of the provided outer `PortPrototype`.

The `shortNames` of `VariableDataPrototypes` and `ParameterDataPrototypes` are used to identify the pair.

[[constr_1071](#)] defines which `PortInterface` elements are compatible depending on the `PortInterface` type and the `swImplPolicy` attributes of the `PortInterface` elements.

- (b) A `VariableAndParameterInterfaceMapping.dataMapping` exists for which the following conditions apply:
 - i. It is (if a corresponding `SwConnector` already exists) referenced by the corresponding `SwConnector`.
 - ii. It references one of the two `VariableDataPrototypes` or `ParameterDataPrototypes` in the role `firstDataPrototype` and the other in the role `secondDataPrototype`.

3. For each such pair, the values of their `isService` attributes are identical.

]()

6.4.3 Connection of Required and Provided Port via PassThroughSwConnector

[constr_1248] Compatibility of PortPrototypes of different DataInterfaces in the context of a PassThroughSwConnector [PortPrototypes of different DataInterfaces are considered compatible if and only if

1. For **at least one** VariableDataPrototype or ParameterDataPrototype defined in the context of the DataInterface of the required outer PortPrototype a compatible VariableDataPrototype or ParameterDataPrototype exists in the DataInterface of the provided outer PortPrototype.

The table 6.1 defines which elements of PortInterface are considered compatible depending on the type of PortInterface as well as the attribute swImplPolicy of the elements of PortInterfaces.

Either the shortName of VariableDataPrototypes and ParameterDataPrototypes are used to identify the pair **or** a PortInterfaceMapping exists that defines which differently named elements of PortInterfaces correlate with each other.

2. For each such pair, the values of the PortInterface.isService attributes are identical.

]()

6.4.4 Compatibility of ParameterDataPrototype and VariableDataPrototype depending on PortInterface Type

Table 6.1 contains a comprehensive description of which combinations of ParameterDataPrototype and VariableDataPrototype used in PortPrototypes typed by various kinds of PortInterfaces are considered compatible.

[constr_1071] compatibility of ParameterDataPrototype and VariableDataPrototype [Combinations of ParameterDataPrototype and VariableDataPrototype used in PortPrototypes typed by various kinds of PortInterfaces shall only be allowed where Table 6.1 contains the value “yes”.]()

The following legend applies for the abbreviations used in table 6.1:

Interface Element i.e. elements of PortInterface

PDP ParameterDataPrototype

VDP VariableDataPrototype

Port Interface i.e. kind of PortInterface

Prm ParameterInterface

S/R SenderReceiverInterface

NvD `NvDataInterface`

Provided Port Required Outer Port Provided Inner Port Required Outer Port			Required Port Required Inner Port Provided Outer Port Provided Outer Port					
PortInterface			Prm			S/R		NvD
Interface Element			PDP			VDP		VDP
SwImplPolicyEnum			fixed	const	standard	standard	queued	standard
Prm	PDP	fixed	yes	yes	yes	yes	no	yes
		const	no	yes	yes	yes	no	yes
		standard	no	no	yes	yes	no	yes
S/R	VDP	standard	no	no	no	yes	no	yes
		queued	no	no	no	no	yes	no
NvD	VDP	standard	no	no	no	yes	no	yes

Table 6.1: Overview of compatibility of `ParameterDataPrototype` and `VariableDataPrototype`

[[constr_1071](#)] defines which `PortInterface` elements are compatible depending on the kind of `PortInterface` and the `swImplPolicy` attributes of the `PortInterface` elements.

[[constr_1287](#)] **Compatibility of `SenderReceiverInterfaces` with respect to `invalidationPolicy`** [`VariableDataPrototypes` defined in the context of the `SenderReceiverInterface` are only compatible if the `invalidationPolicies` have the same value.]()

[[TPS_SWCT_01567](#)] **Default behavior for `invalidationPolicy`** [For `VariableDataPrototypes` and `ParameterDataPrototypes` in the context of `NvDataInterface` respectively `ParameterInterface`, the `invalidationPolicy` is treated like “Invalidation is switched off” (`dontInvalidate`).]([RS_SWCT_00200](#))

6.5 Compatibility of Mode Switch Interfaces

Please note that this compatibility requirement only satisfies static correctness which means that logical consistency is not assured (e.g. that a receiver shall process a certain data value to correctly interpret the following values).

Note that concerning the compatibility of `ModeSwitchInterfaces` it is necessary to distinguish between the context of an `AssemblySwConnector`, the context of an `DelegationSwConnector`, and the context of a `PassThroughSwConnector`.

6.5.1 Connection of Required and Provided Port via `AssemblySwConnector`

Here, the compatibility of `ModeSwitchInterfaces` is considered for the context of an `AssemblySwConnector`.

[constr_1072] Compatibility of `ModeSwitchInterfaces` in the context of an `AssemblySwConnector` [`PortPrototypes` of different `ModeSwitchInterfaces` are compatible if and only if

1. One of the following conditions applies:
 - (a) For the `ModeDeclarationGroupPrototype` defined in the context of the `ModeSwitchInterface` of the required `PortPrototype` a compatible `ModeDeclarationGroupPrototype` exists in the `ModeSwitchInterface` of the provided `PortPrototype`.
 - (b) A `ModeInterfaceMapping.modeMapping` exists for which the following conditions apply:
 - i. It is referenced by the corresponding `SwConnector`.
 - ii. It references one of the two `ModeDeclarationGroupPrototypes` in the role `firstModeGroup` and the other in the role `secondModeGroup`.
2. For each such pair, the values of their `isService` attributes are identical.

]()

6.5.2 Connection of Inner and Outer Port via `DelegationSwConnector`

Here, the compatibility of `ModeSwitchInterfaces` is considered for the context of a `DelegationSwConnector`.

[constr_1073] Compatibility of `ModeSwitchInterfaces` in the context of an `DelegationSwConnector` [`PortPrototypes` of different `ModeSwitchInterfaces` are compatible if and only if

1. One of the following conditions applies:
 - (a) For the `ModeDeclarationGroupPrototype` defined in the context of the `ModeSwitchInterface` of the inner `PortPrototype` a compatible `ModeDeclarationGroupPrototype` exists in the `ModeSwitchInterface` of the outer `PortPrototype`.
 - (b) A `ModeInterfaceMapping.modeMapping` exists for which the following conditions apply:
 - i. It is referenced by the corresponding `SwConnector`.
 - ii. It references one of the two `ModeDeclarationGroupPrototypes` in the role `firstModeGroup` and the other in the role `secondModeGroup`.
2. For each such pair, the values of their `isService` attributes are identical.

]()

6.5.3 Connection of Outer and Outer Port via PassThroughSwConnector

[constr_1249] Compatibility of ModeSwitchInterfaces in the context of a PassThroughSwConnector [PortPrototypes of different ModeSwitchInterfaces are considered compatible if and only if

1. For the ModeDeclarationGroupPrototype defined in the context of the ModeSwitchInterface of the required outer PortPrototype a compatible ModeDeclarationGroupPrototype exists in the ModeSwitchInterface of the provided outer PortPrototype.

Either the shortNames of the ModeDeclarationGroupPrototypes are used to identify the pair or a ModeInterfaceMapping exists that maps the corresponding ModeDeclarationGroupPrototypes.

2. For each such pair, the values of the PortInterface.isService attributes are identical.

]()

6.6 Compatibility of Mode Declaration Group Prototypes

[constr_1074] Compatibility of ModeDeclarationGroupPrototypes [

ModeDeclarationGroupPrototypes are compatible if and only if one of the following conditions applies:

1. They are typed by (read “refer to”) compatible ModeDeclarationGroups.
2. A ModeDeclarationGroupPrototypeMapping exists that identifies the differently named ModeDeclarationGroupPrototypes that correlate with each other. [constr_1210] applies.

]()

6.7 Compatibility of Mode Declaration Groups

[constr_1075] Compatibility of ModeDeclarationGroups [ModeDeclarationGroups are compatible if and only if one of the following conditions applies:

1. All of the following subconditions apply:
 - (a) They define an identical number of ModeDeclarations.
 - (b) Each ModeDeclaration on the required side corresponds to a ModeDeclaration on the provided side with an identical shortName.
 - (c) The initialModes on both sides refer to ModeDeclarations with identical shortNames.

- (d) The attribute `ModeDeclarationGroup.modeUserErrorBehavior.errorReactionPolicy` has identical values on both sides.
- (e) The attribute `ModeDeclarationGroup.modeManagerErrorBehavior.errorReactionPolicy` has identical values on both sides.
- (f) The attribute `ModeDeclarationGroup.modeUserErrorBehavior.defaultMode` either does not exist on both sides or refers on both sides to `ModeDeclarations` with identical `shortNames`.
- (g) The attribute `ModeDeclarationGroup.modeManagerErrorBehavior.defaultMode` either does not exist on both sides or refers on both sides to `ModeDeclarations` with identical `shortNames`.
- (h) one of the following subconditions applies:
 - the attribute `category` has the value `ALPHABETIC_ORDER` on both sides.
 - the attribute `category` has the value `EXPLICIT_ORDER` on both sides **and** the matching `ModeDeclarations` according to 1(b) have the identical values of the attributes `ModeDeclaration.value` **and** also the value of `ModeDeclarationGroup.onTransitionValue` matches on both sides.

2. A `ModeDeclarationMapping` is applied which identifies the corresponding `ModeDeclarations`.

In addition, the compatibility of corresponding `ModeTransitions` shall be checked, i.e. [constr_1194] and [constr_1245] apply. `]()`

[constr_1245] Consideration of `ModeTransitions` for the compatibility of `ModeDeclarationGroups` [One of the following conditions for the consideration of `ModeTransitions` for the compatibility of `ModeDeclarationGroups` shall apply:

- **Either** the mode provider **or** the mode user define `ModeTransitions`.
- The `ModeTransitions` defined in the context of the mode provider are **identical** to the `ModeTransitions` defined in the context of the mode user **or** a `ModeDeclarationMapping` mapping is applied.

`]()`

[constr_1194] Identical `ModeTransitions` [Two `ModeDeclarationGroups` contain identical `modeTransitions` if and only if

1. For each `ModeTransition` defined in the context of the mode provider one `ModeTransition` with the same `shortName` is defined in the context of the mode user.
2. Each pair of `ModeTransitions` in both `ModeDeclarationGroups` identified by their respective `shortName` have identical targets (in terms of the `short-`

Name of the referenced `ModeDeclaration`) of the references `enteredMode` and `exitedMode`.

⌋()

6.8 Compatibility of Argument Prototypes

[constr_1076] Compatibility of `ArgumentDataPrototypes` ⌈ Two `ArgumentDataPrototypes` are compatible if and only if

1. They are typed by compatible `AutosarDataTypes` or a `ClientServerOperationMapping.argumentMapping` exists that references one `ArgumentDataPrototype` in the role `firstDataPrototype` and the other `ArgumentDataPrototype` in the role `secondDataPrototype`.
2. They have the same value of the argument `direction` (`in`, `out` or `inout`), i.e. **[constr_1268]** applies.

⌋()

6.9 Compatibility of Application Errors

[constr_1077] Compatibility of `ApplicationErrors` ⌈ Two `ApplicationErrors` are compatible if and only if one of the following conditions applies:

1. All of the following subconditions apply:
 - (a) They have the same `shortName`.
 - (b) They have the same attributes. Especially the `errorCode` shall be identical in both `ApplicationErrors`.
2. A `ClientServerInterfaceMapping.errorMapping` exists that references one of the `ApplicationErrors` in the role `firstApplicationError` and the other `ApplicationErrors` in the role `secondApplicationError`.

⌋()

6.10 Compatibility of Client/Server Operations

[constr_1078] Compatibility of `ClientServerOperations` ⌈ Two `ClientServerOperations` are compatible if their signatures match. In particular, they are compatible if and only if

1. They have the same number of `ArgumentDataPrototypes`.

2. The n-th arguments of both `ClientServerOperations` are compatible. This implies ordering of `ArgumentDataPrototypes`.
3. They have the same `shortName` (again allows for mapping in `PortInterfaces`).
4. The required `ClientServerOperation` specifies a compatible `ApplicationError` for each `ApplicationError` that is possibly raised by the provided `ClientServerOperation`, maybe more. Thereby, `ClientServerOperations` that refer to a `possibleError` that represents the value `E_OK` are compatible to `ClientServerOperations` that do refer to `possibleErrors` where none of them represents the value `E_OK`.

]()

6.11 Compatibility of Client Server Interfaces

Please note that this compatibility requirement only satisfies static correctness which means that logical consistency is not assured (e.g. that a client shall call a certain operation to allow the server to work correctly).

6.11.1 Connection of Required and Provided Port via AssemblySwConnector

[constr_1079] Compatibility of `ClientServerInterfaces` in the context of an `AssemblySwConnector` [`ClientServerInterfaces` are compatible if and only if

1. One of the following conditions applies:
 - (a) For each `ClientServerOperation` defined in the context of the `ClientServerInterface` of the required `PortPrototype` a compatible `ClientServerOperation` exists in the `ClientServerInterface` of the provided `PortPrototype`. The `shortNames` of `ClientServerOperations` are used to identify the pair.
 - (b) A `ClientServerInterfaceMapping.operationMapping` exists for which the following conditions apply:
 - i. It is referenced by the corresponding `SwConnector`.
 - ii. It references one of the two `ClientServerOperations` in the role `firstOperation` and the other in the role `secondOperation`.
2. For each such pair, the values of their `isService` attributes are identical.

]()

6.11.2 Connection of Inner and Outer Port via DelegationSwConnector

[constr_1080] Compatibility of `ClientServerInterfaces` in the context of an `DelegationSwConnector` [`ClientServerInterfaces` are compatible if and only if

1. One of the following conditions applies:
 - (a) For each `ClientServerOperation` defined in the context of the `ClientServerInterface` of the required inner `PortPrototype` a compatible `ClientServerOperation` exists in the `ClientServerInterface` of the required outer `PortPrototype`. The `shortNames` of `ClientServerOperations` are used to identify the pair.
 - (b) A `ClientServerInterfaceMapping.operationMapping` exists for which the following conditions apply:
 - i. It is referenced by the corresponding `SwConnector`.
 - ii. It references one of the two `ClientServerOperations` in the role `firstOperation` and the other in the role `secondOperation`.
2. One of the following conditions applies:
 - (a) For at least one `ClientServerOperation` defined in the context of the `ClientServerInterface` of the provided inner `PortPrototype` a compatible `ClientServerOperation` exists in the `ClientServerInterface` of the provided outer `PortPrototype`. The `shortNames` of `ClientServerOperations` are used to identify the pair.
 - (b) A `ClientServerInterfaceMapping.operationMapping` exists for which the following conditions apply:
 - i. It is referenced by the corresponding `SwConnector`.
 - ii. It references one of the two `ClientServerOperations` in the role `firstOperation` and the other in the role `secondOperation`.
3. For each such pair, the values of their `isService` attributes are identical.

]()

6.11.3 Connection of Outer and Outer Port via PassThroughSwConnector

[constr_1250] Compatibility of `ClientServerInterfaces` in the context of a `PassThroughSwConnector` [`PortPrototypes` of different `ClientServerInterfaces` are considered compatible if and only if

1. For **at least one** `ClientServerOperation` defined in the context of the `ClientServerInterface` of the provided outer `PortPrototype` a compat-

ible `ClientServerOperation` exists in the `ClientServerInterface` of the required outer `PortPrototype`.

Either the `shortNames` of the `ClientServerOperations` are used to identify the pair or a `ClientServerInterfaceMapping` exists that maps the corresponding `ClientServerOperations`.

2. For each such pair, the values of the `PortInterface.isService` attributes are identical.

]()

6.12 Compatibility of Trigger Interfaces

Please note that this compatibility requirement only satisfies static correctness which means that logical consistency is not assured (e.g. that a client shall call a certain operation to allow the server to work correctly).

6.12.1 Connection of Required and Provided Port via AssemblySwConnector

[constr_1081] Compatibility of `TriggerInterfaces` in the context of an `AssemblySwConnector` [`TriggerInterfaces` are compatible if and only if

1. One of the following conditions applies:
 - (a) For each `Trigger` defined in the context of the `TriggerInterface` of the required `PortPrototype` a compatible `Trigger` exists in the `TriggerInterface` of the provided `PortPrototype`. The `shortNames` of `Trigger` are used to identify the pair.
 - (b) A `TriggerInterfaceMapping.triggerMapping` exists for which the following conditions apply:
 - i. It is referenced by the corresponding `SwConnector`.
 - ii. It references one of the two `Triggers` in the role `firstTrigger` and the other in the role `secondTrigger`.
2. For each such pair, the values of their `isService` attributes are identical.

]()

6.12.2 Connection of Inner and Outer Port via DelegationSwConnector

[constr_1082] Compatibility of `TriggerInterfaces` in the context of an `DelegationSwConnector` [`TriggerInterfaces` are compatible if and only if all of the following conditions apply:

1. One of the following subconditions applies:
 - (a) For each `Trigger` defined in the context of the `TriggerInterface` of the **required** inner `PortPrototype` a compatible `Trigger` exists in the `TriggerInterface` of the **required** outer `PortPrototype`. The `shortNames` of `Trigger` are used to identify the pair.
 - (b) For at least one `Trigger` defined in the context of the `TriggerInterface` of the **provided** outer `PortPrototype` a compatible `Trigger` exists in the `TriggerInterface` of the **provided** inner `PortPrototype`. The `shortNames` of `Trigger` are used to identify the pair.
 - (c) A `TriggerInterfaceMapping.triggerMapping` exists for which all of the following conditions apply:
 - i. It is referenced by the corresponding `SwConnector`.
 - ii. It references one of the two `Triggers` in the role `firstTrigger` and the other in the role `secondTrigger`.
2. For each such pair, the values of their `isService` attributes are identical.

]()

6.12.3 Connection of Outer and Outer Port via PassThroughSwConnector

[constr_1251] Compatibility of `PortPrototypes` of `TriggerInterfaces` in the context of a `PassThroughSwConnector` [`PortPrototypes` of different `TriggerInterfaces` are considered compatible if and only if

1. For **at least one** `Trigger` defined in the context of the `TriggerInterface` of the required outer `PortPrototype` a compatible `Trigger` exists in the `TriggerInterface` of the provided outer `PortPrototype`.

Either the `shortName` of `Triggers` are used to identify the pair **or** a `TriggerInterfaceMapping` exists that that refers to one of the `Triggers` in the role `firstTrigger` and to the other in the role `secondTrigger`.
2. For each such pair, the values of the `PortInterface.isService` attributes are identical.

]()

6.13 Compatibility of Trigger

[constr_1083] Compatibility of `Triggers` [`Triggers` are compatible if they have an identical `shortName`.]()

6.14 Entire Delegation of a Provided Port Prototype

[constr_1084] delegation of a provided outer `PortPrototype` [The delegation of a provided outer `PortPrototype` is properly defined if the following criteria are fulfilled:

1. For each `VariableDataPrototype` or `ParameterDataPrototype` present in the `SenderReceiverInterface`, `NvDataInterface`, or `ParameterInterface` of the provided outer `PortPrototype` at least one connection via `DelegationSwConnector` to a provided inner `PortPrototype` **or** `PassThroughSwConnector` to a required outer `PortPrototype` with a compatible `VariableDataPrototype` or `ParameterDataPrototype` in the `SenderReceiverInterface` `NvDataInterface` or `ParameterInterface` of the provided inner `PortPrototype` **or** required outer `PortPrototype` exists.

Either the `shortNames` of `VariableDataPrototypes` or `ParameterDataPrototypes` are used to identify the pair or a `PortInterfaceMapping` defines which differently named `PortInterface` elements correlate with each other.

Table 6.1 defines which `PortInterface` elements are compatible depending on the kind of `PortInterface` and the `swImplPolicy` attributes of the `PortInterface` elements.

2. For each `VariableDataPrototype` provided by a `PRPortPrototype` that is typed by a `SenderReceiverInterface` or `NvDataInterface` and that is referenced in the role `outerPort` by a `DelegationSwConnector` a corresponding `VariableDataPrototype` owned by an `innerPort` shall be provided by either a `PPortPrototype` or a `PRPortPrototype`.

Either the `shortNames` of `VariableDataPrototypes` are used to identify the pair or a `PortInterfaceMapping` defines which differently named `PortInterface` elements correlate with each other.

3. For the `ModeDeclarationGroupPrototype` present in the `ModeSwitchInterface` of the provided outer `PortPrototype` exactly one connection via `DelegationSwConnector` to a provided inner `PortPrototype` **or** `PassThroughSwConnector` to a required outer `PortPrototype` with a compatible `ModeDeclarationGroupPrototype` in the `ModeSwitchInterface` of the provided inner `PortPrototype` **or** required outer `PortPrototype` exists.

Either the `shortNames` of `ModeDeclarationGroupPrototypes` are used to identify the pair or a `PortInterfaceMapping` defines which differently named `PortInterface` elements correlate with each other.

4. For each `ClientServerOperation` present in the `ClientServerInterface` of the provided outer `PortPrototype` exactly one connection via `DelegationSwConnector` to a provided inner `PortPrototype` **or** `PassThrough`

`SwConnector` to a required outer `PortPrototype` with a compatible `ClientServerOperation` in the `ClientServerInterface` of the provided inner `PortPrototype` **or** required outer `PortPrototype` exists.

Either the `shortNames` of `ClientServerOperations` are used to identify the pair or a `PortInterfaceMapping` defines which differently named `PortInterface` elements correlate with each other.

5. For each `Trigger` present in the `TriggerInterface` of the provided outer `PortPrototype` exactly one connection via `DelegationSwConnector` to a provided inner `PortPrototype` **or** `PassThroughSwConnector` to a required outer `PortPrototype` with a compatible `Trigger` in the `TriggerInterface` of the provided inner `PortPrototype` **or** required outer `PortPrototype` exists.

Either the `shortNames` of `Triggers` are used to identify the pair or a `PortInterfaceMapping` defines which differently named `PortInterface` elements correlate with each other.

]()

6.14.1 Split and Merge of PortInterface Elements

With the definition of compatibility rules in chapter 6.4, 6.11, and 6.12 it is possible to split and distribute elements of a `PortPrototype` of type of a `PortInterface` containing a superset of `PortInterface` elements to `PortPrototypes` of type of `PortInterfaces` containing subsets of `PortInterface` elements.

Please find examples that explain the usage of splitting and merging in section 6.16.2.

6.15 Compatibility in Case of a Flat ECU Extract

Please note that in the case of a flat ECU extract of software-components specific compatibility rules apply. To some extent, these rules contradict the rules existing for the pure VFB approach (see chapter 6). That is, if the split-and-merge pattern has been applied on the creation of `DelegationSwConnectors` it might happen that compatibility rules defined in chapter 6 are violated.

However, given that the flattened ECU extract has been created out of a valid `CompositionSwComponentType` the flattened ECU extract does not become invalid in this case. In other words, the transformation does not create an invalid model out of a valid model.

However, to support this statement it is necessary to define additional compatibility rules that properly cover this case and allow for a successful validation of the flattened ECU extract.

For the flat ECU extract the compatibility of `SenderReceiverInterfaces`, `NvDataInterfaces`, and `ParameterInterfaces` is considered for connecting of `PortPrototypes` with a `DelegationSwConnector`.

[constr_1085] Compatibility in the case of a flat ECU extract [`PortPrototypes` of different `SenderReceiverInterfaces`, `NvDataInterfaces`, and `ParameterInterfaces` are compatible if and only if for at least one `VariableDataPrototype` or `ParameterDataPrototype` defined in the context of the `SenderReceiverInterface`, `NvDataInterface`, or `ParameterInterface` of the `RPortPrototype` a compatible `VariableDataPrototype` or `ParameterDataPrototype` exists in the `SenderReceiverInterface`, `NvDataInterface`, or `ParameterInterface` of the provided `PortPrototype`.

The compatibility of `PortInterface` elements depends on the kind of `PortInterface` and the `swImplPolicy` attributes of the `PortInterface` elements.

Either the `shortNames` of `VariableDataPrototypes` and `ParameterDataPrototypes` are used to identify the pair or a `PortInterfaceMapping` defines which differently named `PortInterface` elements correlate with each other. `]()`

For clarification, table 6.1 defines which `PortInterface` elements are compatible depending on the kind of `PortInterface` and the `swImplPolicy` attributes of the `PortInterface` elements.

Please note that in case of the flat ECU extract it might happen that `AssemblySwConnectors` that connect to a specific `RPortPrototype` also connect to `PPortPrototypes` that do not fulfill the compatibility rule specified in 6.4.1.

In particular, the `dataElements` might correspond to `dataElements` defined in the scope of different `PPortPrototypes`. In other words, in the flat ECU extract it is possible to merge `dataElements` from different providers.

6.16 Compatibility Examples

This section provides some examples that may explain the compatibility of `PortPrototypes`.

6.16.1 Compatibility on Assembly Level

The rules for compatibility with respect to the connection of `dataElements` by means of `AssemblySwConnectors` are perhaps easier to digest than the delegation case but nonetheless it seems appropriate to provide a set of examples that illustrate the compatibility issue.

6.16.1.1 Legal Use

One of the less trivial examples of this kind is the case of sender/receiver n:1 communication. Figure 6.1 sketches a case where both sender software-components provide the full set of `dataElements` that are required by the `RPortPrototype` of the receiving software-component.

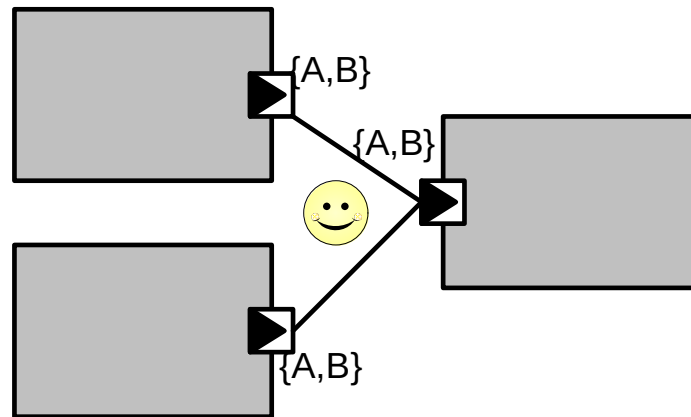


Figure 6.1: legal n:1 communication

The next case (exemplified by Figure 6.2) implements a situation where one sender provides two `dataElements` {A,b} while the other sender provides only as subset of these, i.e. {B}.

As the `RPortPrototype` of the receiving software-component requires only the `dataElement` {B} compatibility issues will not occur because for every required `dataElement` a compatible `dataElement` is provided.

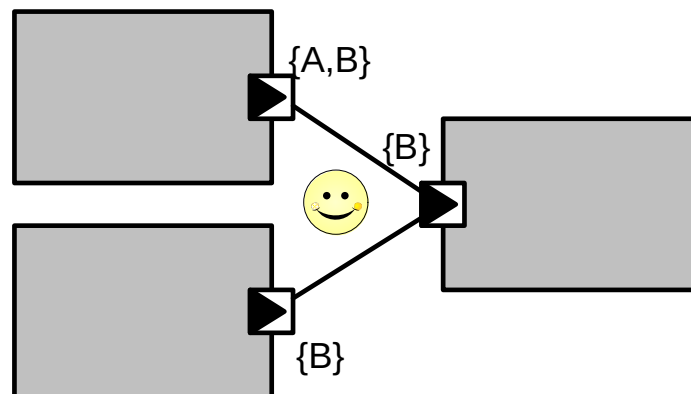


Figure 6.2: legal n:1 communication

6.16.1.2 Illegal Use

One possible example for an illegal configuration of a sender/receiver communication is the scenario sketched in Figure 6.3. Although the sender software-components in total

provide the set of required `dataElements` the *individual* `AssemblySwConnectors` create incompatible connections between sender and receiver.

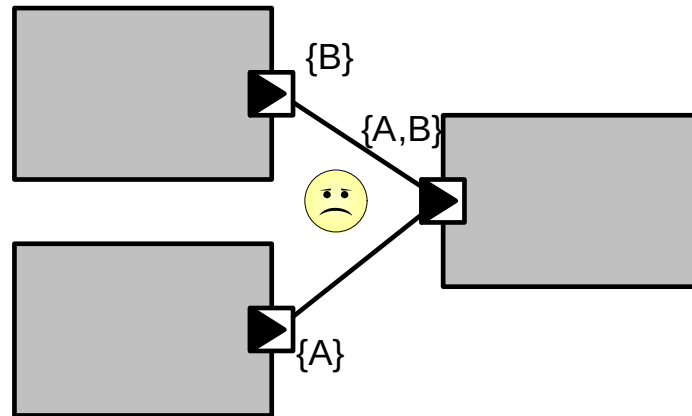


Figure 6.3: illegal n:1 communication

6.16.2 Compatibility on Delegation Level

The rules for compatibility with respect to the delegation of `dataElements` perhaps require some explanation in terms of examples. The first example 6.4 describes a legal situation where two `DelegationSwConnectors` split the `dataElements` contained in the `RPortPrototype` owned by a `CompositionSwComponentType`.

6.16.2.1 Legal Use

The examples explain the usage of `DelegationSwConnectors` in different configurations and different values of `DelegatedPortAnnotation`. Please note that the `DelegatedPortAnnotation` is usually defined before the internal structure of a `CompositionSwComponentType` is fully clarified.

At a later point in time it has to be consistent or can be removed. Decorating the example with applicable values of `DelegatedPortAnnotation` should facilitate the understanding of the meaning of the `DelegatedPortAnnotation`.

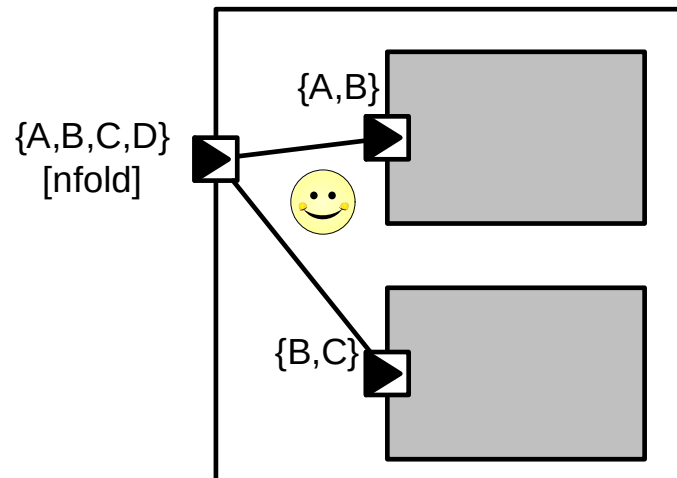


Figure 6.4: Legal split of delegation connector

All required `dataElements` are provided by the `DelegationSwConnectors` attached to the delegation `RPortPrototype`. The fact that `dataElement` D is not conveyed to any of the `RPortPrototypes` owned by the `SwComponentPrototypes` does not have any impact on the compatibility.

In other words: the `RPortPrototype` at the `CompositionSwComponentType` actually contains the superset of `dataElements` {A ,B, C, D}. The two required inner `PortPrototypes` of the `SwComponentPrototypes` contain the subsets of `VariableDataPrototypes` {A, B} and {B, C}. In this case the resulting communication pattern on the VFB for B would be 1:n.

This requires the value of the attribute `signalFan` of `DelegatedPortAnnotation` to be set to the value `nfold`.

In the next example the `RPortPrototype` of the `CompositionSwComponentType` contains the superset of `dataElements` {A ,B}. The two `RPortPrototypes` of the `SwComponentPrototypes` contain *different* subsets, i.e. {A} and {B}.

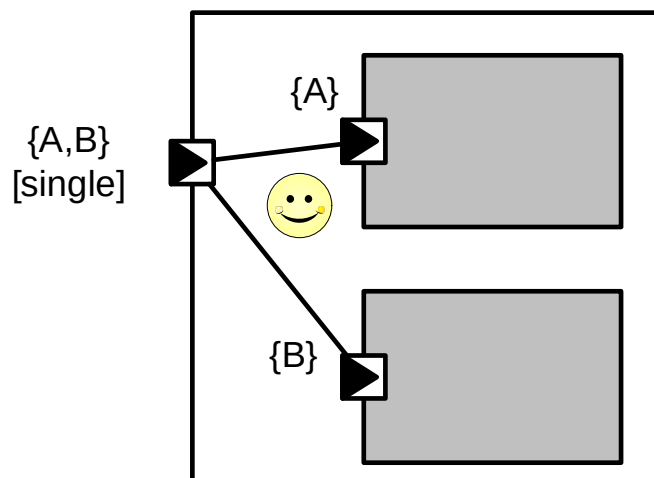


Figure 6.5: Legal split of delegation connector

In this case the resulting communication pattern on the VFB would be $n:1$. In this case the value of the attribute `signalFan` of `DelegatedPortAnnotation` should be set to `single`.

The next example is about the merge of `DelegationSwConnectors`. The `PPort-Prototype` owned by the `CompositionSwComponentType` contains a superset of `dataElements` $\{A, B\}$. The two `PPortPrototypes` of the `SwComponentPrototypes` contain a *disjoint* subset each, i.e. $\{A\}$ and $\{B\}$.

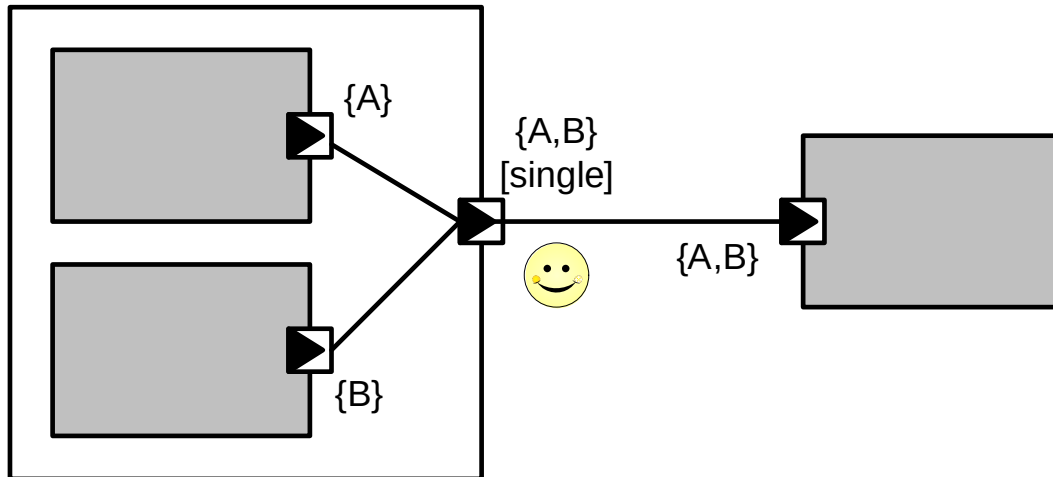


Figure 6.6: Legal merge of delegation connector

In this case the resulting communication pattern on the VFB would be $1:x$, with x taking values between 0 and n . In this case the value of the attribute `signalFan` of `DelegatedPortAnnotation` should be set to `single`. All `VariableDataPrototypes` of the provided outer `PortPrototypes` are provided by exactly one provided inner `PortPrototype`.

As a variation of this theme, the next example features a `PPortPrototype` owned by a `CompositionSwComponentType` that contains the superset of `dataElements` $\{A, B, C\}$.

The `PPortPrototypes` of the `SwComponentPrototypes` in turn contain subsets of `dataElements`, i.e. $\{A, B\}$ and $\{B, C\}$. In this case the resulting communication pattern on the VFB for $\{B\}$ would be $n:1$.

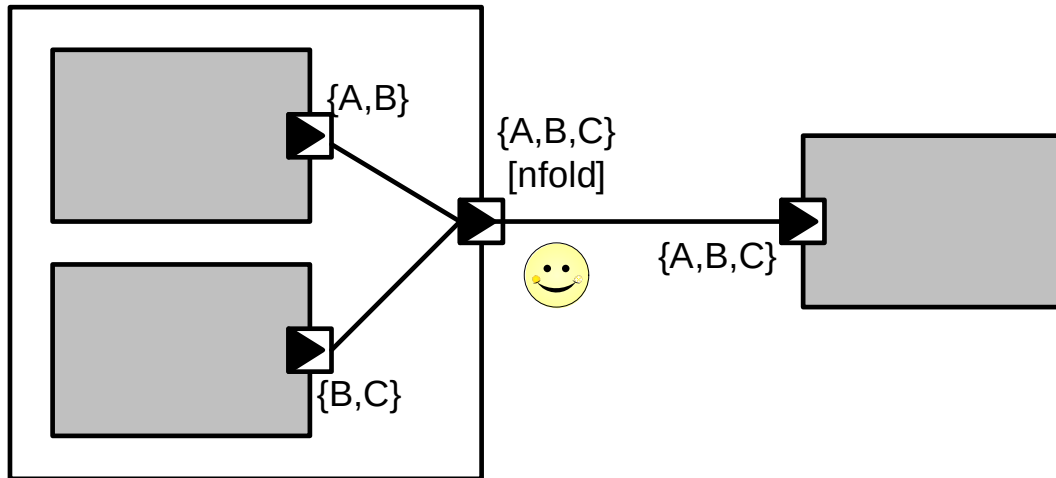


Figure 6.7: Legal merge of delegation connector

This would require the value of the attribute `signalFan` of `DelegatedPortAnnotation` to be set to `nfold`. All `dataElements` of the delegation `PPortPrototype` are provided by at least one `PPortPrototype` of the `SwComponentPrototypes`. Therefore the criteria of `entire delegation` defined in chapter 6.14 are fulfilled.

The next example looks very similar. However, the subtle difference is that the second `SwComponentPrototype` provides `dataElements` {C,D} rather than {B,C}.

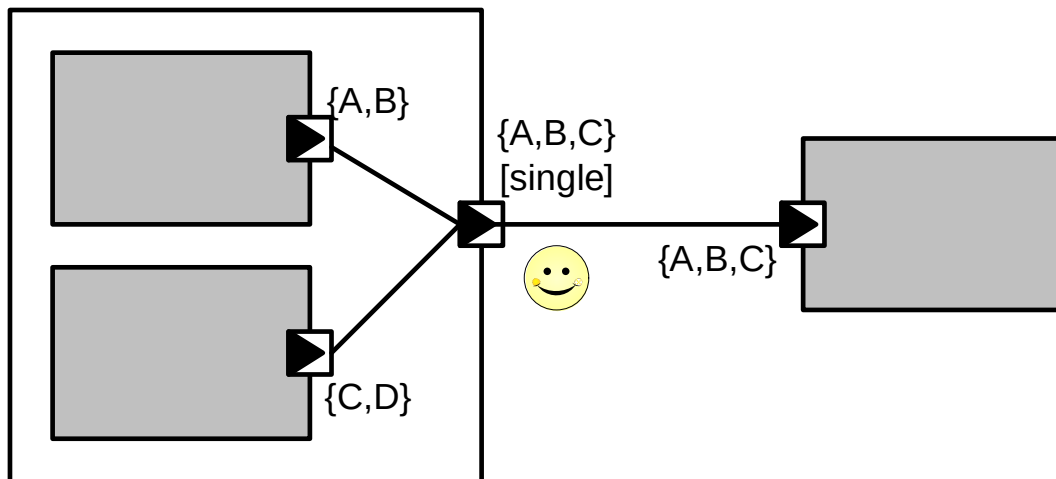


Figure 6.8: Legal merge of delegation connector

Although `dataElement` {D} does not appear in the delegation `PPortPrototype` the compatibility rules are fully satisfied with this scenario.

The next example shows a valid delegation of `SwConnectors` that goes end-to-end via `CompositionSwComponentTypes` to included `SwComponentPrototypes`.

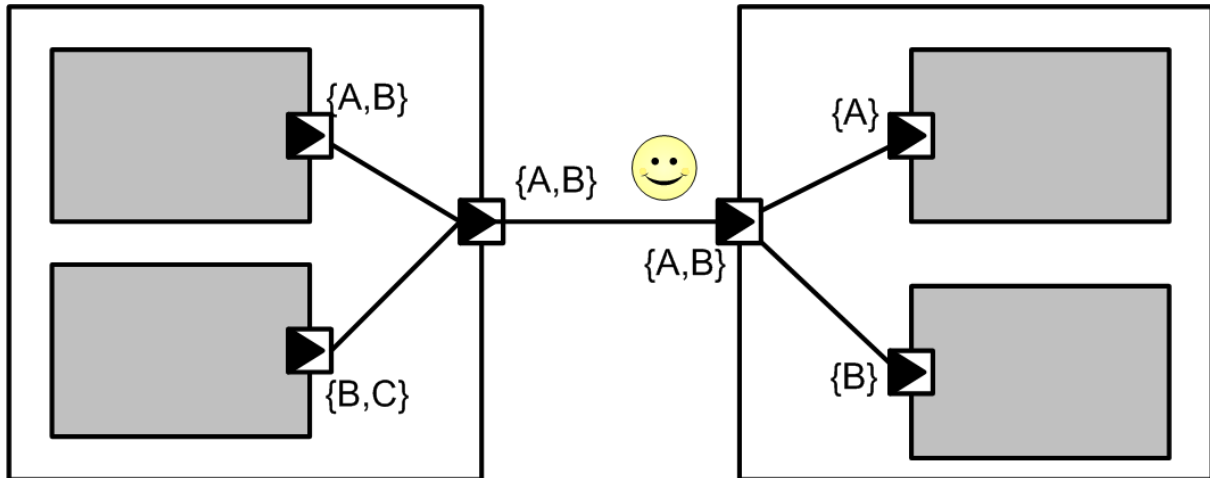


Figure 6.9: Valid delegation of **SwConnectors** that goes end-to-end

6.16.2.2 Illegal Use

The first example for an illegal use of splitting of **dataElements** suffers from the fact that not all **dataElements** owned by the **RPortPrototypes** of the **SwComponentPrototypes** are available from the connected **RPortPrototypes** owned by the **CompositionSwComponentType**.

Although **dataElements** the connections in total match ({A} and {B} are connected to a **PortPrototype** requiring {A,B}) the compatibility rules are not fulfilled because they apply separately for *each SwConnector*

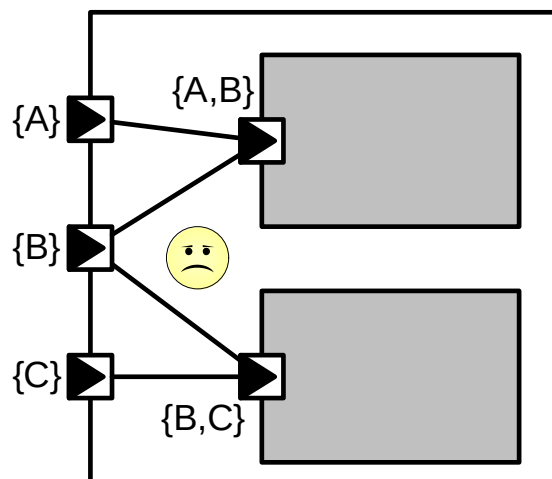


Figure 6.10: Illegal split of delegation connector

In the next example compatibility is also not fulfilled because the required **dataElement** {E} is not provided by the delegation **RPortPrototype**.

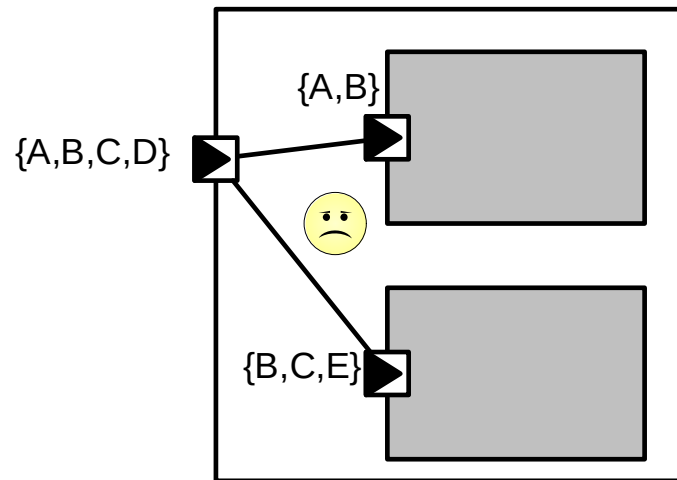


Figure 6.11: Illegal split of delegation connector

An incompatible merge of [DelegationSwConnectors](#) is sketched in Figure 6.12. In this case the [dataElement](#) {E} is *not* provided by one of the [PPortPrototypes](#) owned by the [SwComponentPrototypes](#) inside the [CompositionSwComponentType](#).

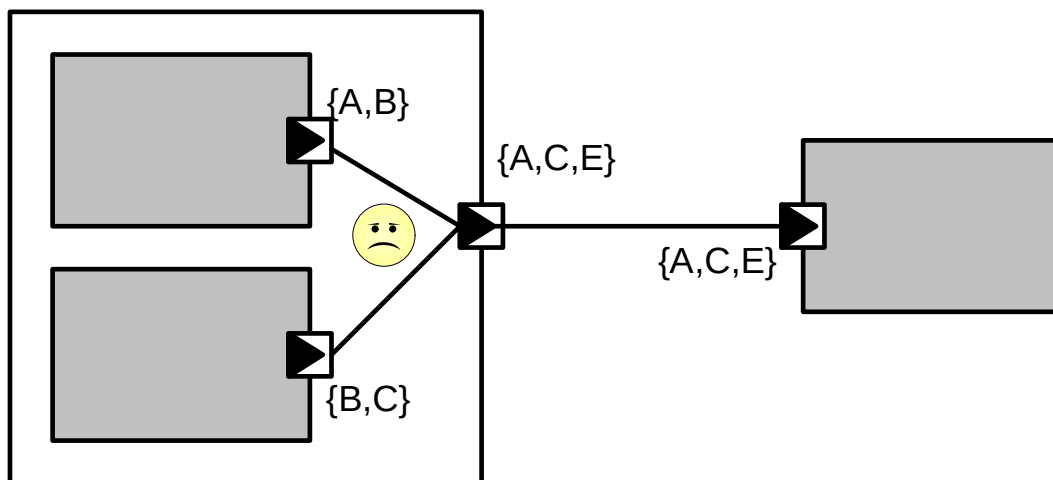


Figure 6.12: Illegal merge of delegation connector

The next example shows an invalid delegation of [SwConnectors](#) that goes end-to-end via [CompositionSwComponentTypes](#) to included [SwComponentPrototypes](#).

Similar to the example sketched in Figure 6.12, the [dataElement](#) {E} is *not* provided by one of the [PPortPrototypes](#) owned by the [SwComponentPrototypes](#) inside the [CompositionSwComponentType](#).

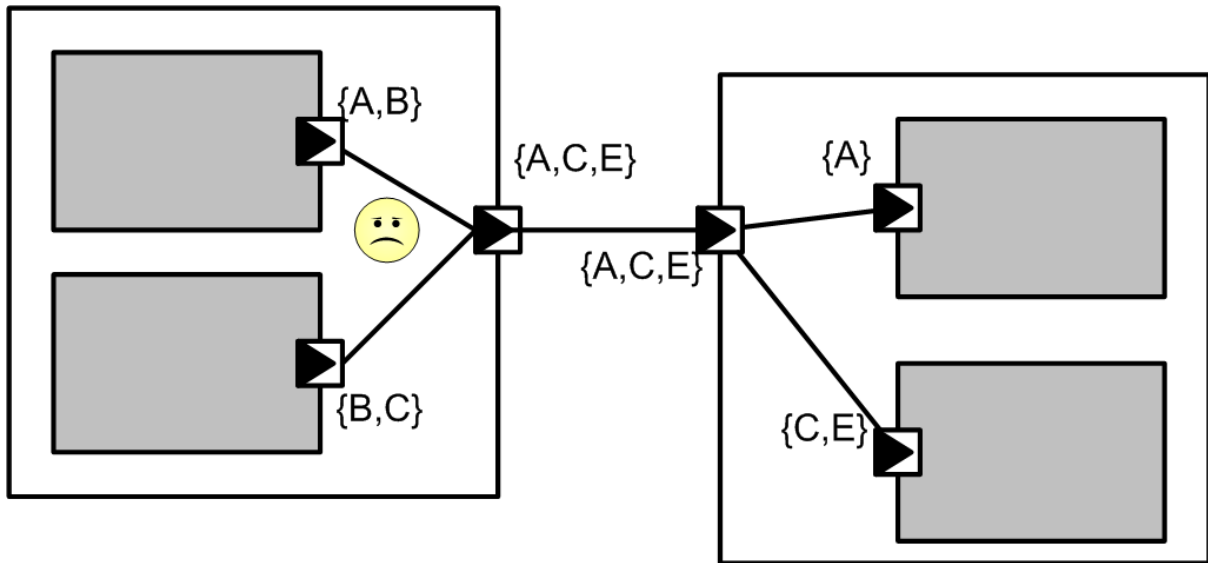


Figure 6.13: Invalid delegation of **SwConnectors** that goes end-to-end

7 Internal Behavior

7.1 Introduction

[TPS_SWCT_01075] **SwcInternalBehavior** [SwcInternalBehavior provides means for formally defining the behavior of an AtomicSwComponentType.](RS_SWCT_03040)

This chapter focuses on the description of the SwcInternalBehavior meta-class and the various meta-classes it aggregates. An overview of the meta-class is sketched in Figure 7.2. Please note that SwcInternalBehavior inherits from InternalBehavior.

The role of SwcInternalBehavior in the context of an AUTOSAR software-component is depicted in Figure 7.1. As mentioned in section 3.2, the reason to make the aggregation of SwcInternalBehavior to AtomicSwComponentType «atpVariation,atpSplittable» is to allow for the development of SwcInternalBehavior in a later process step (e.g. after the VFB view has been completed).

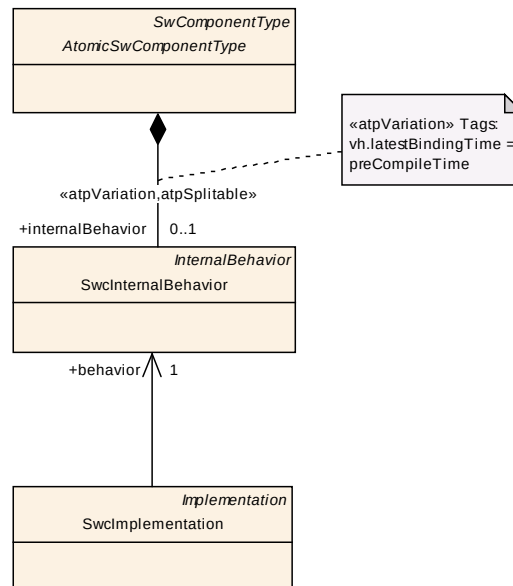


Figure 7.1: The “big picture” of SwcInternalBehavior

Class	SwcInternalBehavior			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior			
Note	The SwcInternalBehavior of an AtomicSwComponentType describes the relevant aspects of the software-component with respect to the RTE, i.e. the RunnableEntities and the RTEEvents they respond to.			
Base	ARObject,AtpClassifier,AtpFeature,AtpStructureElement,Identifiable,Internal Behavior,MultilanguageReferrable,Referrable			
Attribute	Datatype	Mul.	Kind	Note

Attribute	Datatype	Mul.	Kind	Note
arTypedPerInstanceMemory	VariableDataPrototype	*	aggr	Defines an AUTOSAR typed memory-block that needs to be available for each instance of the SW-component. This is typically only useful if supportsMultipleInstantiation is set to "true" or if the component defines NVRAM access via permanent blocks. The aggregation of arTypedPerInstanceMemory is subject to variability with the purpose to support variability in the software components implementations. Typically different algorithms in the implementation are requiring different number of memory objects. Stereotypes: atpSplitable; atpVariation Tags: atp.Splitkey=shortName, variation Point.shortLabel vh.latestBindingTime=preCompileTime
event	RTEEvent	*	aggr	This is a RTEEvent specified for the particular SwcInternalBehavior. The aggregation of RTEEvent is subject to variability with the purpose to support the conditional existence of RTE events. Note: the number of RTE events might vary due to the conditional existence of PortPrototypes using DataReceivedEvents or due to different scheduling needs of algorithms. Stereotypes: atpSplitable; atpVariation Tags: atp.Splitkey=shortName, variation Point.shortLabel vh.latestBindingTime=preCompileTime
explicitInterRunnableVariable	VariableDataPrototype	*	aggr	Implement state message semantics for establishing communication among runnables of the same component. The aggregation of explicitInterRunnableVariable is subject to variability with the purpose to support variability in the software components implementations. Typically different algorithms in the implementation are requiring different number of memory objects. Stereotypes: atpSplitable; atpVariation Tags: atp.Splitkey=shortName, variation Point.shortLabel vh.latestBindingTime=preCompileTime
handleTerminationAndRestart	HandleTerminationAndRestartEnum	1	attr	This attribute controls the behavior with respect to stopping and restarting. The corresponding AtomicSwComponentType may either not support stop and restart, or support only stop, or support both stop and restart.

Attribute	Datatype	Mul.	Kind	Note
implicitInterRunnableVariable	VariableDataPrototype	*	aggr	Implement state message semantics for establishing communication among runnables of the same component. The aggregation of implicitInterRunnableVariable is subject to variability with the purpose to support variability in the software components implementations. Typically different algorithms in the implementation are requiring different number of memory objects. Stereotypes: atpSplitable; atpVariation Tags: atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime
includedDataTypeSet	IncludedDataTypeSet	*	aggr	The includedDataTypeSet is used by a software component for its implementation. Stereotypes: atpSplitable Tags: atp.Splitkey=includedDataTypeSet
includedModeDeclarationGroupSet	IncludedModeDeclarationGroupSet	*	aggr	This aggregation represents the included ModeDeclarationGroups Stereotypes: atpSplitable Tags: atp.Splitkey=includedModeDeclarationGroupSet
instantiationDataDefProps	InstantiationDataDefProps	*	aggr	The purpose of this is that within the context of a given SwComponentType some data def properties of individual instantiations can be modified. The aggregation of InstantiationDataDefProps is subject to variability with the purpose to support the conditional existence of PortPrototypes and component local memories like "perInstanceParameter" or "arTypedPerInstanceMemory". Stereotypes: atpSplitable; atpVariation Tags: atp.Splitkey=instantiationDataDefProps, variationPoint.shortLabel vh.latestBindingTime=preCompileTime
perInstanceMemory	PerInstanceMemory	*	aggr	Defines a per-instance memory object needed by this software component. The aggregation of PerInstanceMemory is subject to variability with the purpose to support variability in the software components implementations. Typically different algorithms in the implementation are requiring different number of memory objects. Stereotypes: atpSplitable; atpVariation Tags: atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime

Attribute	Datatype	Mul.	Kind	Note
perInstanceParameter	ParameterDataPrototype	*	aggr	Defines parameter(s) or characteristic value(s) that needs to be available for each instance of the software-component. This is typically only useful if supportsMultipleInstantiation is set to "true". The aggregation of perInstanceParameter is subject to variability with the purpose to support variability in the software components implementations. Typically different algorithms in the implementation are requiring different number of memory objects. Stereotypes: atpSplitable; atpVariation Tags: atp.Splitkey=shortName, variation Point.shortLabel vh.latestBindingTime=preCompileTime
portAPIOption	PortAPIOption	*	aggr	Options for generating the signature of port-related calls from a runnable to the RTE and vice versa. The aggregation of PortPrototypes is subject to variability with the purpose to support the conditional existence of ports. Stereotypes: atpSplitable; atpVariation Tags: atp.Splitkey=portAPIOption, variation Point.shortLabel vh.latestBindingTime=preCompileTime
runnable	RunnableEntity	1..*	aggr	This is a RunnableEntity specified for the particular SwcInternalBehavior. The aggregation of RunnableEntity is subject to variability with the purpose to support the conditional existence of RunnableEntities. Note: the number of RunnableEntities might vary due to the conditional existence of PortPrototypes using DataReceivedEvents or due to different scheduling needs of algorithms. Stereotypes: atpSplitable; atpVariation Tags: atp.Splitkey=shortName, variation Point.shortLabel vh.latestBindingTime=preCompileTime

Attribute	Datatype	Mul.	Kind	Note
serviceDependency	SwcServiceDependency	*	aggr	Defines the requirements on AUTOSAR Services for a particular item. The aggregation of SwcServiceDependency is subject to variability with the purpose to support the conditional existence of ports as well as the conditional existence of ServiceNeeds. The SwcServiceDependency owned by an SwcInternalBehavior can be located in a different physical file in order to support that SwcServiceDependency might be provided in later development steps or even by different expert domain (e.g OBD expert for Obd related Service Needs) tools. Therefore the aggregation is «atpSplitable». Stereotypes: atpSplitable; atpVariation Tags: atp.Splitkey=shortName, variation Point.shortLabel vh.latestBindingTime=preCompileTime
sharedParameter	ParameterDataPrototype	*	aggr	Defines parameter(s) or characteristic value(s) shared between SwComponentPrototypes of the same SwComponentType The aggregation of sharedParameter is subject to variability with the purpose to support variability in the software components implementations. Typically different algorithms in the implementation are requiring different number of memory objects. Stereotypes: atpSplitable; atpVariation Tags: atp.Splitkey=shortName, variation Point.shortLabel vh.latestBindingTime=preCompileTime
supportsMultipleInstantiation	Boolean	1	attr	Indicate whether the corresponding software-component can be multiply instantiated on one ECU. In this case the attribute will result in an appropriate component API on programming language level (with or without instance handle).
variationPointProxy	VariationPointProxy	*	aggr	Proxy of a variation points in the C/C++ implementation. Stereotypes: atpSplitable Tags: atp.Splitkey=shortName

Table 7.1: SwcInternalBehavior

Enumeration	HandleTerminationAndRestartEnum
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior
Note	Controls the behavior of an AtomicSwComponentType with respect to stop and restart.
Literal	Description

canBeTerminated	Supports termination.
canBeTerminatedAndRestarted	Supports termination and restarting.
noSupport	Stop and restart is not supported at all.

Table 7.2: HandleTerminationAndRestartEnum

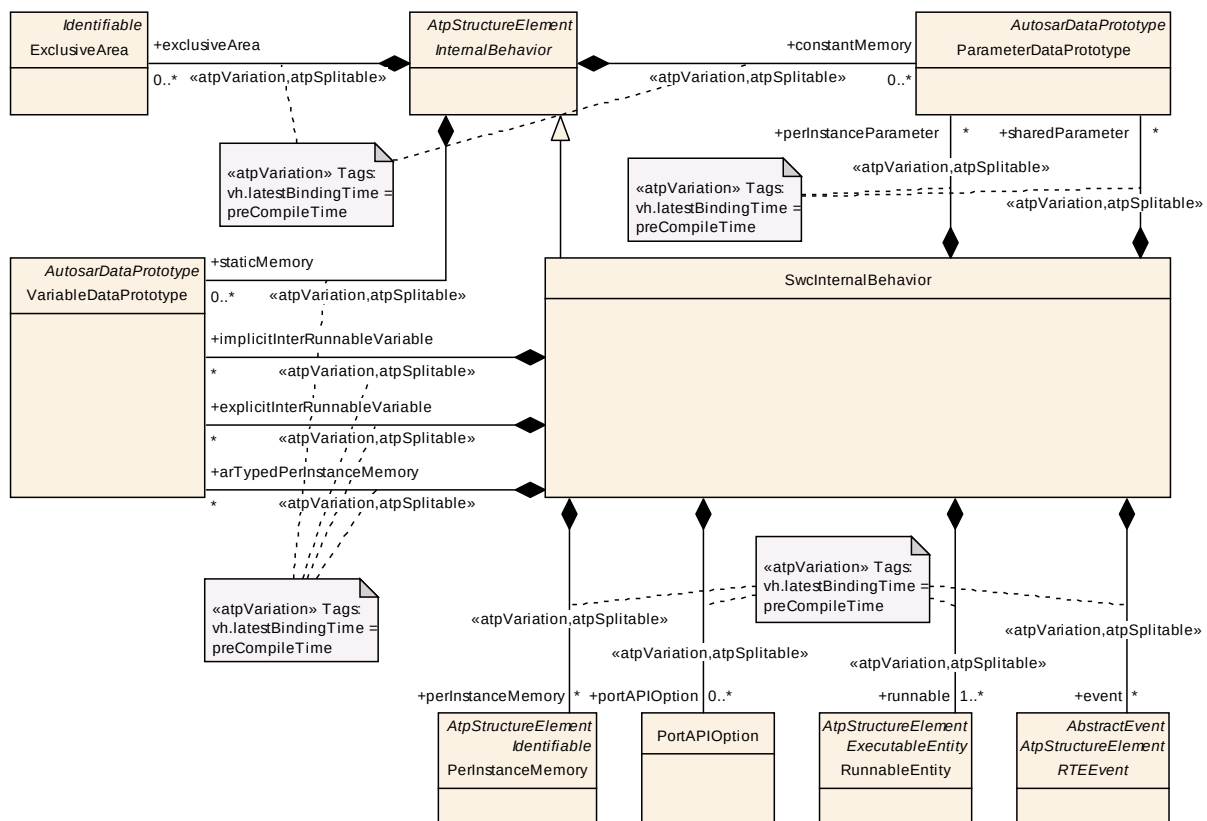


Figure 7.2: SwcInternalBehavior

7.2 Runnable Entity

The concept of **RunnableEntity** (more details can be found in Figure 7.3) is defined in the specification of the Virtual Function Bus [3].

[TPS_SWCT_01030] RunnableEntity [**RunnableEntity**s are the smallest code-fragments that are provided by a software-component and are (at least indirectly) a subject for scheduling by the underlying operating system.] (**RS_SWCT_00070**, **RS_SWCT_00090**, **RS_SWCT_03050**)

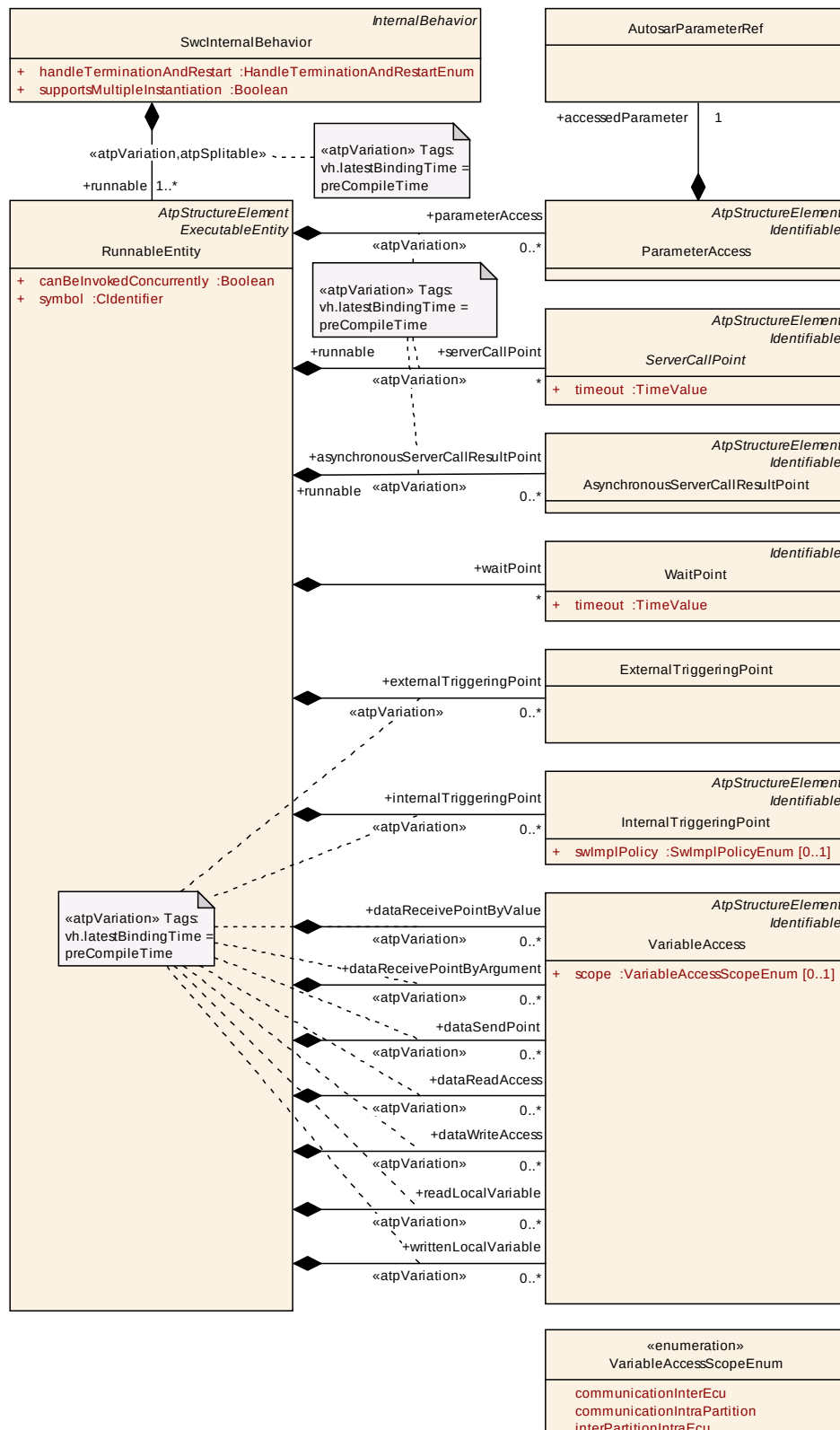


Figure 7.3: Details of RunnableEntity

[TPS_SWCT_01097] CompositionSwComponentType cannot have RunnableEntities [It is intentionally not possible for CompositionSwComponent-

Type to define a `SwcInternalBehavior`. Consequently, `CompositionSwComponentTypes` don't have `RunnableEntities` by themselves.]([RS_SWCT_00070](#), [RS_SWCT_00090](#), [RS_SWCT_03050](#))

[TPS_SWCT_01098] Only `AtomicSwComponentType` can have `RunnableEntities` [Only the `AtomicSwComponentType` that are populating a `CompositionSwComponentType` as `SwComponentPrototypes` may have `RunnableEntities`.]([RS_SWCT_00070](#), [RS_SWCT_00090](#), [RS_SWCT_03050](#))

This correlation is depicted in Figure 7.4.

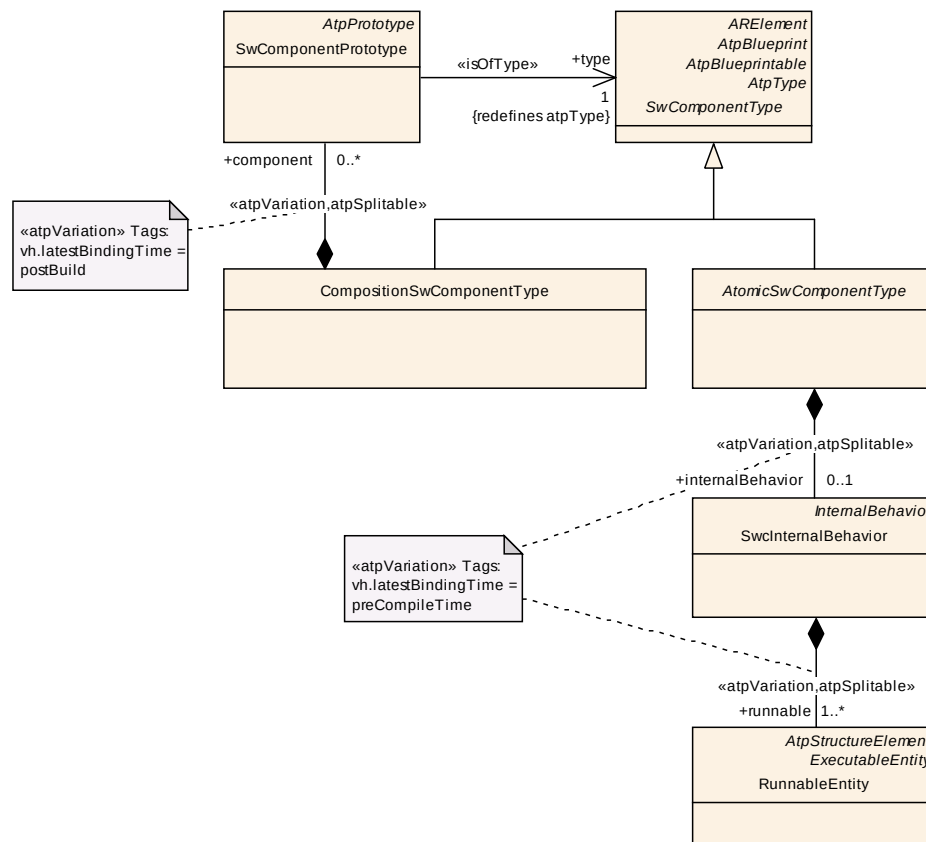


Figure 7.4: Only `AtomicSwComponentTypes` may have `RunnableEntities`

Please note that `RunnableEntities` exist in several categories that have different properties. Please find more explanation about categories of `RunnableEntities` in section 7.2.4.4.

Class	RunnableEntity			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior			
Note	A RunnableEntity represents the smallest code-fragment that is provided by an AtomicSwComponentType and are executed under control of the RTE. RunnableEntities are for instance set up to respond to data reception or operation invocation on a server.			
Base	ARObject, AtpClassifier, AtpFeature, AtpStructureElement, ExecutableEntity , Identifiable , MultilanguageReferrable , Referrable			
Attribute	Datatype	Mul.	Kind	Note
argument (ordered)	RunnableEntityArgument	*	aggr	This represents the formal definition of a an argument to a RunnableEntity.
asynchronousServerCallResultPoint	AsynchronousServerCallResultPoint	*	aggr	The server call result point admits a runnable to fetch the result of an asynchronous server call. The aggregation of AsynchronousServerCallResultPoint is subject to variability with the purpose to support the conditional existence of client server PortPrototypes and the variant existence of server call result points in the implementation. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime
canBeInvokedConcurrently	Boolean	1	attr	If the value of this attribute is set to "true" the enclosing RunnableEntity can be invoked concurrently (even for one instance of the corresponding AtomicSwComponentType). This implies that it is the responsibility of the implementation of the RunnableEntity to take care of this form of concurrency. Note that the default value of this attribute is set to "false".
dataReadAccess	VariableAccess	*	aggr	RunnableEntity has implicit read access to dataElement of a sender-receiver PortPrototype or nv data of a nv data PortPrototype. The aggregation of dataReadAccess is subject to variability with the purpose to support the conditional existence of sender receiver ports or the variant existence of dataReadAccess in the implementation. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime

Attribute	Datatype	Mul.	Kind	Note
dataReceivePointByArgument	VariableAccess	*	aggr	RunnableEntity has explicit read access to dataElement of a sender-receiver PortPrototype or nv data of a nv data PortPrototype. The result is passed back to the application by means of an argument in the function signature. The aggregation of dataReceivePointByArgument is subject to variability with the purpose to support the conditional existence of sender receiver PortPrototype or the variant existence of data receive points in the implementation. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime
dataReceivePointByValue	VariableAccess	*	aggr	RunnableEntity has explicit read access to dataElement of a sender-receiver PortPrototype or nv data of a nv data PortPrototype. The result is passed back to the application by means of the return value. The aggregation of dataReceivePointByValue is subject to variability with the purpose to support the conditional existence of sender receiver ports or the variant existence of data receive points in the implementation. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime
dataSendPoint	VariableAccess	*	aggr	RunnableEntity has explicit write access to dataElement of a sender-receiver PortPrototype or nv data of a nv data PortPrototype. The aggregation of dataSendPoint is subject to variability with the purpose to support the conditional existence of sender receiver PortPrototype or the variant existence of data send points in the implementation. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime
dataWriteAccess	VariableAccess	*	aggr	RunnableEntity has implicit write access to dataElement of a sender-receiver PortPrototype or nv data of a nv data PortPrototype. The aggregation of dataWriteAccess is subject to variability with the purpose to support the conditional existence of sender receiver ports or the variant existence of dataWriteAccess in the implementation. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime

Attribute	Datatype	Mul.	Kind	Note
externalTriggeringPoint	ExternalTriggeringPoint	*	aggr	The aggregation of ExternalTriggeringPoint is subject to variability with the purpose to support the conditional existence of trigger ports or the variant existence of external triggering points in the implementation. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime
internalTriggeringPoint	InternalTriggeringPoint	*	aggr	The aggregation of InternalTriggeringPoint is subject to variability with the purpose to support the variant existence of internal triggering points in the implementation. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime
modeAccessPoint	ModeAccessPoint	*	aggr	The runnable has a mode access point. The aggregation of ModeAccessPoint is subject to variability with the purpose to support the conditional existence of mode ports or the variant existence of mode access points in the implementation. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime
modeSwitchPoint	ModeSwitchPoint	*	aggr	The runnable has a mode switch point. The aggregation of ModeSwitchPoint is subject to variability with the purpose to support the conditional existence of mode ports or the variant existence of mode switch points in the implementation. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime
parameterAccess	ParameterAccess	*	aggr	The presence of a ParameterAccess implies that a RunnableEntity needs read only access to a ParameterDataPrototype which may either be local or within a PortPrototype. The aggregation of ParameterAccess is subject to variability with the purpose to support the conditional existence of parameter ports and component local parameters as well as the variant existence of ParameterAccess (points) in the implementation. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime

Attribute	Datatype	Mul.	Kind	Note
readLocalVariable	VariableAccess	*	aggr	The presence of a readLocalVariable implies that a RunnableEntity needs read access to a VariableDataPrototype in the role of implicitInterRunnableVariable or explicitInterRunnableVariable. The aggregation of readLocalVariable is subject to variability with the purpose to support the conditional existence of implicitInterRunnableVariable and explicitInterRunnableVariable or the variant existence of readLocalVariable (points) in the implementation. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime
serverCallPoint	ServerCallPoint	*	aggr	The RunnableEntity has a ServerCallPoint. The aggregation of ServerCallPoint is subject to variability with the purpose to support the conditional existence of client server PortPrototypes or the variant existence of server call points in the implementation. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime
symbol	CIdentifier	1	ref	The symbol describing this RunnableEntity's entry point. This is considered the API of the RunnableEntity and is required during the RTE contract phase.
waitPoint	WaitPoint	*	aggr	The WaitPoint associated with the RunnableEntity.
writtenLocalVariable	VariableAccess	*	aggr	The presence of a writtenLocalVariable implies that a RunnableEntity needs write access to a VariableDataPrototype in the role of implicitInterRunnableVariable or explicitInterRunnableVariable. The aggregation of writtenLocalVariable is subject to variability with the purpose to support the conditional existence of implicitInterRunnableVariable and explicitInterRunnableVariable or the variant existence of writtenLocalVariable (points) in the implementation. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime

Table 7.3: RunnableEntity

[TPS_SWCT_01302] Semantics of [minimumStartInterval](#) [The attribute [ExecutableEntity.minimumStartInterval](#) defines the time interval that the RTE will guarantee to not go below between scheduling two consecutive executions of the corresponding [RunnableEntity](#).]()

[TPS_SWCT_01303] `symbol` attribute describes the `RunnableEntity`'s entry point [The `RunnableEntity.symbol` attribute is describing the `RunnableEntity`'s entry point.]()

The implication `RunnableEntity.symbol` on the uniqueness of symbols in the scope of one `EcuInstance` is described in [constr_2025] [11].

A `RunnableEntity` inherits several attributes from its base class `ExecutableEntity` due to the fact that these are also used in the Basic Software Module Description Template [7]. Here the following constraint applies:

[constr_4082] `RunnableEntity.reentrancyLevel` shall not be set. [The optional attribute `reentrancyLevel` shall not be set for a `RunnableEntity`. This attribute would define more specific reentrancy features than the mandatory attribute `canBeInvokedConcurrently`. These features are currently only supported for Basic Software.]()

Please note that the formal definition of the semantics of a `RunnableEntity` has strong relations to the specification of the AUTOSAR RTE [2]. The definition of the RTE semantics, however, is not in the scope of this document.

However, the formal definition requires some background discussion that can't be completely left out of this document. Otherwise the meaning of specific model elements could not be understood properly.

7.2.1 Concurrency and Reentrancy of a `RunnableEntity` that cannot be Invoked Concurrently

This section applies to the case that the value of the attribute `canBeInvokedConcurrently` is `false`. During runtime, each `RunnableEntity` of each instance of an `AtomicSwComponentType` is in a specific run-time state.

The details of the definition and semantics of run-time states can be found in [2]. Nevertheless, this chapter contains a brief description of the fundamental concepts in order to properly being able to discuss the formal modeling of `RunnableEntities`.

[TPS_SWCT_01313] Conditions for a transition from `suspended` to `to be started` [The `SwcInternalBehavior` describes for each `RunnableEntity` the conditions for a transition from `suspended` to `to be started` should occur. This is done using the concept of an `RTEEvent`.]()

When a `RunnableEntity` is in state `to be started`, the RTE can decide to start running the `RunnableEntity`. The delay between entering the state `to be started` (e.g. a message has been received in response to which the `RunnableEntity` should run) and moving into the state `running` (the first instruction of the `RunnableEntity` has been executed) depends on the scheduling strategy of the RTE, i.e. the mapping of `RunnableEntities` on AUTOSAR OS tasks.

The transition from the state `running` into the state `suspended` is in the hands of the `RunnableEntity`: the transition occurs when the `RunnableEntity` returns (thereby handing over control to the AUTOSAR OS [28]). Some `RunnableEntities` (like cat. 2 `RunnableEntities`) might never return to the `suspended` state once they entered the `running` state.

They might enter the `preempted` state when being preempted. The same applies if a `RunnableEntity` needs to wait for a `WaitPoint` to be unblocked.

[TPS_SWCT_01304] Cat. 1A and 1B `RunnableEntities` will eventually terminate
[Cat. 1A and 1B `RunnableEntities` will eventually return after having executed a specific finite algorithm (the execution time of which might be provided).]()

[TPS_SWCT_01305] `RunnableEntity` as one that cannot be invoked concurrently
[In case the `SwcInternalBehavior` defines a `RunnableEntity` as one that cannot be invoked concurrently it is the responsibility of the RTE to make sure that the `RunnableEntity` is never started concurrently (for example, in two different AUTOSAR OS tasks). This implies that the implementation of the `AtomicSwComponentType` does not need to worry about concurrency issues.]()

For example: The internal behavior of an `AtomicSwComponentType` `MyComponentType` describes a `RunnableEntity` `R1` which should be enabled when an operation on a client-server p-port of the `AtomicSwComponentType` is invoked. The `AtomicSwComponentType` specifies that the `RunnableEntity` `R1` cannot be invoked concurrently.

The `AtomicSwComponentType` `MyComponentType` is instantiated on an ECU. When a call of the operation is received, the corresponding instance of the `RunnableEntity` `R1` is enabled and the RTE will start executing the `RunnableEntity` (the `RunnableEntity` is in state `running`) in a task eventually managed by the AUTOSAR OS.

If another call of the operation is received while the `RunnableEntity` is in state `running` it is not allowed that the RTE runs the `RunnableEntity` again in a second task. Rather, the RTE has to wait (and maybe queue the second incoming request) until the `RunnableEntity` has returned and has moved to the `suspended` state.

7.2.2 Concurrency and Reentrancy of a `RunnableEntity` that can be Invoked Concurrently

This section applies to the case that the value of the attribute `canBeInvokedConcurrently` is set to `true`.

In this case, it is allowed that the same `RunnableEntity` is running several times concurrently in different AUTOSAR OS tasks. This implies that the state machine defined in [2] is not the state of the `RunnableEntity` any more, but can be cloned an arbitrary number of times.

[TPS_SWCT_01306] Software-component description itself does not put any bounds on the number of concurrent invocations of a `RunnableEntity` [The software-component description itself does not put any bounds on the number of concurrent invocations of the `RunnableEntity` that are allowed.

The software-component description only specifies whether the `RunnableEntity` can be invoked concurrently or not.

Allowing concurrent invocation of a `RunnableEntity` implies that the implementation of the `AtomicSwComponentType` needs to take care of this additional form of concurrency. `()`

For example: The `SwcInternalBehavior` of a component-type `MyComponentType` describes a `RunnableEntity` `R1` which should be enabled when a `ClientServerOperation` on a `PPortPrototype` typed by a `ClientServerInterface` of the `AtomicSwComponentType` is invoked.

The `AtomicSwComponentType` specifies that the `RunnableEntity` `R1` can be invoked concurrently. The `AtomicSwComponentType` `MyComponentType` is instantiated on an ECU.

When a call of the `ClientServerOperation` is received the corresponding instance of the `RunnableEntity` `R1` is enabled and the RTE will start executing the `RunnableEntity` (the `RunnableEntity` is in state `running`) in a task eventually managed by the AUTOSAR OS.

If another call of the `ClientServerOperation` is received, it is allowed that the same `RunnableEntity` is started again in a different task.

A typical use-case of concurrent `RunnableEntity`s is the implementation of AUTOSAR services. The AUTOSAR services will typically take care of concurrency internally: several software-components can directly use the services in parallel.

The ECU-integrator could then decide that the `RunnableEntity` implementing the AUTOSAR service runs directly in the context (in the task) of the `AtomicSwComponentType` invoking the service.

This is a very efficient and direct coupling between the client and the server: the connector between the client and the server is reduced to a local function-call.

7.2.3 Timed Activation of Runnable Entities

In many cases, `RunnableEntity`s need to be activated in response to timing events rather than related to communication (e.g. the reception of a response to an asynchronous operation invocation). Many `RunnableEntity`s will need to run cyclically with a fixed rate.

The approach taken in the software-component description is to define so-called `TimingEvents` (please find more details in Figure 7.5) as special kinds of `RTEEvents`.

So far, only one kind of timing-related `RTEEvent` has been defined: a simple periodic `TimingEvent`.

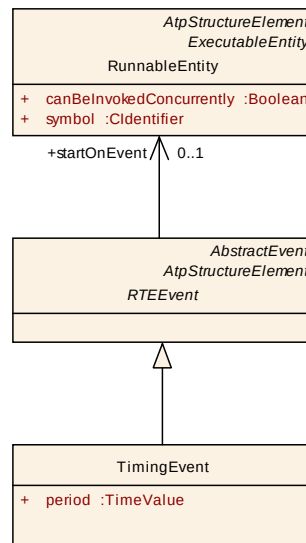


Figure 7.5: Periodic activation of RunnableEntities

[TPS_SWCT_01519] RTE executes certain RunnableEntity periodically [If the `SwcInternalBehavior` of an `AtomicSwComponentType` requires that the RTE executes certain `RunnableEntity`s periodically, the description needs to define a `TimingEvent` with the desired period.

This `TimingEvent` then contains a reference to the Runnable that needs to be executed with this period.]()

7.2.4 Additional Remarks and Clarifications

7.2.4.1 Reentrancy and Multiple Instantiation

This chapter is emphasizing on the specific meanings of combinations of the attributes `SwcInternalBehavior.supportsMultipleInstantiation` and `RunnableEntity.canBeInvokedConcurrently`.

[TPS_SWCT_01307] supportsMultipleInstantiation vs. canBeInvokedConcurrently [The semantics of combining the attributes `supportsMultipleInstantiation` and `canBeInvokedConcurrently` is summarized in Table 7.4.]()

supportsMultipleInstantiation	canBeInvokedConcurrently	Implication for an implementation of a <code>RunnableEntity</code>
false	false	This implies that the implementation of the <code>RunnableEntity</code> will never be invoked concurrently from several tasks. The implementation does not need to care about reentrancy issues and can typically use static variables to store state.

true	false	In case there are several instances of the same <code>AtomicSwComponentType</code> on the local ECU, the implementation of the <code>RunnableEntity</code> can still be invoked concurrently from several tasks. However, there will be no concurrent invocations of the implementation with the same <code>instance handle</code> . To ensure that this is safe, the implementation will typically use per-instance memory.
false/true	true	In this case the <code>RunnableEntity</code> can be invoked concurrently from several tasks, even with the same <code>instance handle</code> .

Table 7.4: supportsMultipleInstantiation vs. canBeInvokedConcurrently

In case the implementation of a `AtomicSwComponentType` decides to map several `RunnableEntities` to the same `symbol` there are reentrancy problems to be sorted out. However, this scenario is not supported by RTE [2] anyway and shall therefore be avoided.

7.2.4.2 Reentrancy and “Library Functions”

Note that all code that is called by different `RunnableEntities` (like e.g. library routines, etc.) shall obviously be reentrant. A filter algorithm implemented in C, for example, is not allowed to store values from previous runs by means of static variables or variables with external binding.

7.2.4.3 Compatibility of ClientServerOperations triggering the same RunnableEntity

[TPS_SWCT_01309] signature of a `RunnableEntity` depends on the connected RTEEvent [The signature of a `RunnableEntity` depends on the connected `RTEEvent`. Multiple `OperationInvokedEvents` are only supported if all referred `ClientServerOperations` would result in the same `RunnableEntity` signature for the server `RunnableEntity`.]()

[constr_2000] Compatibility of `ClientServerOperations` triggering the same `RunnableEntity` [The `ClientServerOperations` are considered compatible if the number of arguments (which can be `ArgumentDataPrototypes` or related `PortDefinedArgumentValues`) is equal and the corresponding arguments (i.e. first argument on both sides, second argument on both sides, etc.) are compatible.

In particular, this means that:

- for combinations of `ArgumentDataPrototypes` and `ArgumentDataPrototypes` where the `serverArgumentImplPolicy` is set to `useArgumentType` the referred `ImplementationDataTypes` shall be compatible.

In case of data types of `category` `STRUCTURE` all by order matching `ImplementationDataTypeElements` shall be named equally.

- for combinations of `PortDefinedArgumentValues` and `ArgumentDataPrototypes` where the `serverArgumentImplPolicy` is set to `useArgumentType` the referred `ImplementationDataTypes` shall be compatible.
- for combinations of `ArgumentDataPrototypes` and `ArgumentDataPrototypes` where the `serverArgumentImplPolicy` is set to `useArrayBaseType` the referred `ImplementationDataTypes` of category `ARRAY` shall have compatible `ImplementationDataTypeElements`.

In case of `ImplementationDataTypeElements` of category `STRUCTURE` all by order matching `ImplementationDataTypeElements` of the structure shall be named equally.

- for `ArgumentDataPrototypes` where the `serverArgumentImplPolicy` is set to `useVoid` an arbitrary `ImplementationDataType` is referred to.

In addition, it is required that the **return value defined on both sides shall match** (in terms of `Std_ReturnType` vs. `void`) and also the `possibleErrors` are compatible.
 ⌋()

[TPS_SWCT_01520] Implication of the existence of `possibleError` on compatibility of `ClientServerOperations` ⌈ An implication of [constr_2000] is that a `ClientServerOperation` that defines **any** `possibleError` is **not** compatible with a `ClientServerOperation` that defines **no** `possibleError` at all because this configuration leads to different data type of the return value of the C function that implements the applicable `RunnableEntity`. ⌋()

7.2.4.4 Categories of Runnable Entities

[TPS_SWCT_01310] Categories of `RunnableEntity`s ⌈ `RunnableEntity`s are subdivided into the following categories:

Category 1

Category 1 `RunnableEntity`s do not have `WaitPoints` and are required to terminate in a finite amount of time. Category 1 is divided into two subcategories: Category 1A and Category 1B. Category 1A `RunnableEntity`s are only allowed to use implicit API's. Category 1B `RunnableEntity`s are additionally allowed to invoke a server and use explicit API's.

Category 2

In contrast to Category 1 `RunnableEntity`s, `RunnableEntity`s of category 2 always aggregate at least one `WaitPoint`, for more details see Figure 7.3¹. Typically, such a `RunnableEntity` implements an internal loop where one iteration through the loop is triggered whenever a `WaitPoint` is resolved. ⌋()

¹Category 2 `RunnableEntity`s usually have to be mapped to *Extended Tasks*, because only extended tasks provide the task state `WAITING`.

7.2.4.5 Arguments of a Runnable Entity

In many cases an RTE generator will be able to figure out not only the number and data type of arguments to a `RunnableEntity` but also the name of the arguments. In some cases, however, formal support from the upstream templates is required to facilitate this task.

[TPS_SWCT_01311] Name of an operation argument [This support is available by means of the meta-class `RunnableEntityArgument` that contributes the name of the argument by means of the value of the attribute `symbol`.]()

As a `RunnableEntity` might need to define many arguments the aggregation of `RunnableEntityArgument` at `RunnableEntity` in the role `argument` has the multiplicity `0..*` and as the order of these arguments is significant the meta-model defines the aggregation as `ordered`².]()

[constr_1164] Number of arguments owned by a RunnableEntity [If a given `RunnableEntity` owns `RunnableEntityArguments` in the role `argument`, then the number of these `RunnableEntityArguments` shall be identical to the number of applicable `portArgValues` of the `PortAPIOption` that references the `PortPrototype` that in turn is referenced by the `OperationInvokedEvent` that references the `RunnableEntity` **plus** the number of `ArgumentDataPrototypes` aggregated in the role `argument` by the `ClientServerOperation` referenced by said `OperationInvokedEvent`.]()

[constr_1165] Applicability of RunnableEntityArgument [The existence of a `RunnableEntityArgument` is limited to `RunnableEntities` triggered by a `ClientServerOperation`.]()

[TPS_SWCT_01312] RunnableEntity has a mapping to BswModuleEntry [The existence of `RunnableEntityArguments` in the role `argument` owned by a `RunnableEntity` shall be **ignored** by an RTE generator if a mapping to a `BswModuleEntry` exists. In this case the name of arguments to the `RunnableEntity` shall be derived from the applicable `SwServiceArgs` owned by the mapped `BswModuleEntry`.]()

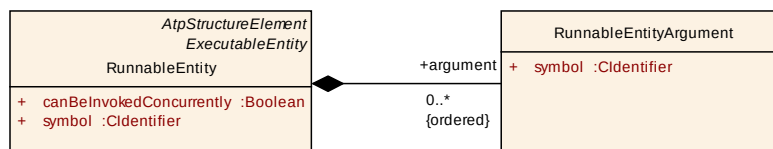


Figure 7.6: Arguments of a RunnableEntity

²as the arguments are **ordered** they do not need to be `Referrable` in order to be able to identify individual `arguments`

Class	RunnableEntityArgument			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::RunnableEntity			
Note	This meta-class represents the ability to provide specific information regarding the arguments to a RunnableEntity.			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
symbol	CIdentifier	1	ref	This represents the symbol to be generated into the actual signature on the level of the C programming language.

Table 7.5: RunnableEntityArgument

7.2.5 Activation Reason of a Runnable Entity

It is feasible to activate a given [RunnableEntity](#) by means of several [RTEEvents](#). In many cases, it is therefore necessary to retrieve the information about the activating [RTEEvent](#) from within the implementation of the [RunnableEntity](#).

As a typical use case, consider a [RunnableEntity](#) that is cyclically activated (by means of a [TimingEvent](#)) and in addition it shall also be executed sporadically, e.g. in response to the reception ([DataReceivedEvent](#)) of a [dataElement](#).

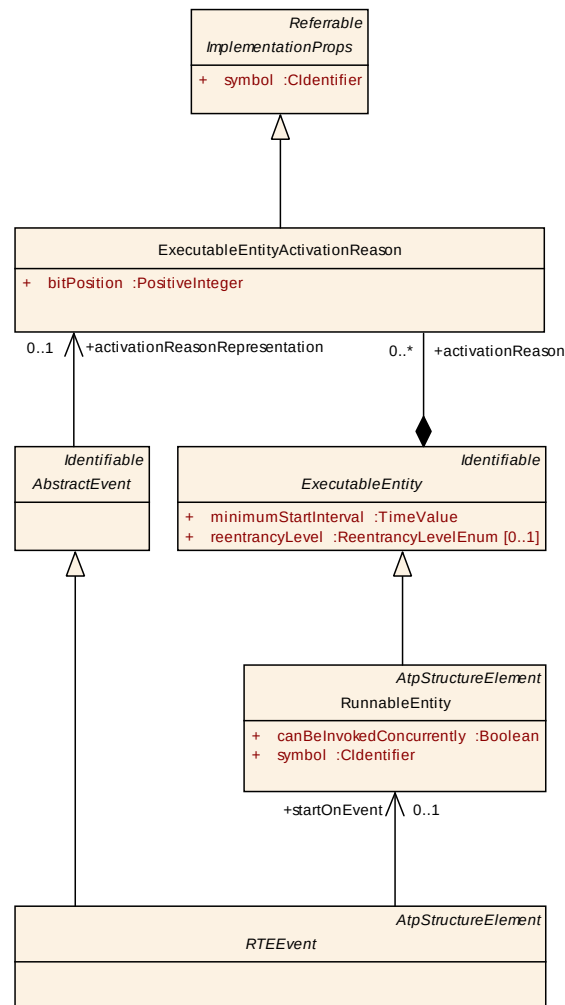


Figure 7.7: **ExecutableEntityActivationReason** and **RunnableEntity**

[TPS_SWCT_01469] RTE API for retrieving the current activation reason [The aggregation of a **ExecutableEntityActivationReason** allows for the RTE generator to create an RTE API for retrieving the current activation reason.]([RS_SWCT_03045](#))

For details about the implementation of this feature, please refer to the specification of the RTE [2]

[constr_1226] Applicable range for **ExecutableEntityActivationReason.bitPosition** [The value of attribute **ExecutableEntityActivationReason.bitPosition** shall be in the range of 0 .. 31.]()

[constr_1227] Value of attribute **ExecutableEntityActivationReason.bitPosition shall be unique** [The value of attributes **ExecutableEntityActivationReason.bitPosition** and **ExecutableEntityActivationReason.symbol** shall be unique in the context of the enclosing **RunnableEntity**.]()

[constr_1228] **RTEEvent that is referenced by a **WaitPoint** in the role **trigger** shall not reference **ExecutableEntityActivationReason**** [An **RTEEvent**

that is referenced by a `WaitPoint` in the role `trigger` shall not reference `ExecutableEntityActivationReason` in the role `activationReasonRepresentation`.]()

The rationale for the existence of [constr_1228] is obviously that in the described situation the `RunnableEntity` is already activated and therefore the mentioned `RTEEvent` does not deliver any information related to the activation reason of said `RunnableEntity`.

Class	ExecutableEntityActivationReason			
Package	M2::AUTOSARTemplates::CommonStructure::InternalBehavior			
Note	This meta-class represents the ability to define the reason for the activation of the enclosing ExecutableEntity.			
Base	ARObject, ImplementationProps, Referrable			
Attribute	Datatype	Mul.	Kind	Note
bitPosition	PositiveInteger	1	attr	This attribute allows for defining the position of the enclosing ExecutableEntityActivationReason in the activation vector.

Table 7.6: ExecutableEntityActivationReason

Please note that the attribute `ExecutableEntityActivationReason.symbol` is needed for the generation of a unique identifier that represents the specific activation reason in the RTE code.

7.2.6 Runnable Entity for Initialization Purpose

One way to make sure that certain initializations are applied before a software-component enters its state of normal operation is to use the AUTOSAR mode-management, in particular by defining a `ModeDeclarationGroup` that contains a specific `ModeDeclaration` with the semantics of representing a mode that is exclusively used for setting up and initializing a software-component.

However, this approach comes with a certain amount of footprint that may be acceptable in some cases but there may also be cases where a simpler approach comes in handy. The simple approach to initialization consists of a `RunnableEntity` that is triggered by a special kind of `RTEEvent`, i.e. the so-called `InitEvent`.

[TPS_SWCT_01525] `InitEvent` references a `RunnableEntity` in the role `startOnEvent` [In addition to using a mode-based approach for executing initialization `RunnableEntities` it is also possible to let an `InitEvent` reference a `RunnableEntity` in the role `startOnEvent`.

This approach to the initialization of software-components is orthogonal to the mode-based approach. Especially, the `RunnableEntities` triggered by an `InitEvent` are expected to be executed after the RTE has been fully initialized. This means restrictions regarding the availability of RTE APIs during the ECU initialization are not relevant for `RunnableEntities` triggered by an `InitEvent`.]([RS_SWCT_03290](#))

[constr_1257] No `WaitPoints` allowed [A `RunnableEntity` referenced by an `InitEvent` in the role `startOnEvent` shall not aggregate a `WaitPoint`.]()

Rationale: a `WaitPoint` may indefinitely defer the completion of the `RunnableEntities` triggered by an `InitEvent` and therefore contradict the semantics of the `RunnableEntity`.

[constr_1258] Value of `minimumStartInterval` for `RunnableEntities` triggered by an `InitEvent` [The value of the attribute `ExecutableEntity.minimumStartInterval` for a `RunnableEntities` that is triggered by an `InitEvent` shall always be set to 0.]()

Rationale: it does not make sense to talk about intervals of activating `RunnableEntities` triggered by an `InitEvent` as these are not supposed to be executed repeatedly.

[constr_1259] Aggregation of `AsynchronousServerCallPoint` and `AsynchronousServerCallResultPoint` [A `RunnableEntity` referenced by an `InitEvent` in the role `startOnEvent` may aggregate an `AsynchronousServerCallPoint` but it shall not aggregate an `AsynchronousServerCallResultPoint`.]()

Rationale: as mentioned before `WaitPoints` shall not be aggregated by a `RunnableEntities` triggered by an `InitEvent` in the role `startOnEvent`. It is allowed (although considered unlikely to happen) to have an `AsynchronousServerCallPoint` but it is not allowed to fetch the result of the call within the same `RunnableEntity`.

A `RunnableEntity` triggered by an `InitEvent` in the role `startOnEvent` may aggregate a `SynchronousServerCallPoint` but the usage of this configuration is discouraged.

[constr_1260] No mode disabling for `InitEvents` [An `InitEvent` shall not have a reference to a `ModeDeclaration` in the role `disabledMode`.]()

Rationale: the concept of `RunnableEntity` triggered by an `InitEvent` is (as mentioned before) orthogonal to the mode concept and therefore shall be implemented independent of modes.

7.3 RTEEvent

During execution, several `RTEEvents` will occur, such as the reception of a remote invocation of a `ClientServerOperation` on a `PPortPrototype` or a timeout on an `RPortPrototype` that is not receiving the `VariableDataPrototypes` it expects to receive.

[TPS_SWCT_01314] `RTEEvent` [The description of an `RTEEvent` includes two aspects:

1. defining an [RTEEvent](#)
2. defining how the RTE should deal with the [RTEEvent](#) when it occurs.

]()

Class	AbstractEvent (abstract)			
Package	M2::AUTOSARTemplates::CommonStructure::InternalBehavior			
Note	This meta-class represents the abstract ability to model an event that can be taken to implement application software or basic software in AUTOSAR.			
Base	ARObject, Identifiable , MultilanguageReferrable, Referrable			
Attribute	Datatype	Mul.	Kind	Note
activationReasonRepresentation	ExecutableEntityActivationReason	0..1	ref	If the activationReasonRepresentation is referenced from the enclosing AbstractEvent this shall be taken as an indication that the latter contributes to the activating vector of this ExecutableEntity that owns the referenced ExecutableEntityActivationReason.

Table 7.7: AbstractEvent

Class	RTEEvent (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::RTEEvents			
Note	Abstract base class for all RTE-related events			
Base	ARObject, AbstractEvent , AtpClassifier, AtpFeature, AtpStructureElement, Identifiable , MultilanguageReferrable, Referrable			
Attribute	Datatype	Mul.	Kind	Note
disabledMode	ModeDeclaration	*	iref	Reference to the Modes that disable the Event. Stereotypes: atpSplitable Tags: atp.Splitkey=contextPort, contextModeDeclarationGroupPrototype, targetModeDeclaration
startOnEvent	RunnableEntity	0..1	ref	RunnableEntity starts when the corresponding RTEEvent occurs.

Table 7.8: RTEEvent

Class	AsynchronousServerCallReturnsEvent			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::RTEEvents			
Note	This event is raised when an asynchronous server call is finished.			
Base	ARObject, AbstractEvent , AtpClassifier, AtpFeature, AtpStructureElement, Identifiable , MultilanguageReferrable, RTEEvent , Referrable			
Attribute	Datatype	Mul.	Kind	Note
eventSource	AsynchronousServerCallResultPoint	1	ref	The referenced AsynchronousServerCallResultPoint which is raises the RTEEvent in case of returning asynchronous server call.

Attribute	Datatype	Mul.	Kind	Note

Table 7.9: AsynchronousServerCallReturnsEvent

Class	DataSendCompletedEvent			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::RTE Events			
Note	The event is raised when the referenced data elements have been sent or an error occurs.			
Base	ARObject, AbstractEvent , AtpClassifier, AtpFeature, AtpStructure Element, Identifiable , MultilanguageReferrable, RTEEvent , Referrable			
Attribute	Datatype	Mul.	Kind	Note
eventSource	VariableAccess	1	ref	The variable access that triggers the event.

Table 7.10: DataSendCompletedEvent

Class	DataWriteCompletedEvent			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::RTE Events			
Note	This event is raised if an implicit write access was successful or an error occurred.			
Base	ARObject, AbstractEvent , AtpClassifier, AtpFeature, AtpStructure Element, Identifiable , MultilanguageReferrable, RTEEvent , Referrable			
Attribute	Datatype	Mul.	Kind	Note
eventSource	VariableAccess	1	ref	The variable access that triggers the event.

Table 7.11: DataWriteCompletedEvent

Class	DataReceivedEvent			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::RTE Events			
Note	The event is raised when the referenced data elements are received.			
Base	ARObject, AbstractEvent , AtpClassifier, AtpFeature, AtpStructure Element, Identifiable , MultilanguageReferrable, RTEEvent , Referrable			
Attribute	Datatype	Mul.	Kind	Note
data	VariableDataPrototype	0..1	iref	Data element referenced by event

Table 7.12: DataReceivedEvent

Class	DataReceiveErrorEvent			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::RTE Events			
Note	This event is raised by the RTE when the Com layer detects and notifies an error concerning the reception of the referenced data element.			
Base	ARObject, AbstractEvent , AtpClassifier, AtpFeature, AtpStructure Element, Identifiable , MultilanguageReferrable, RTEEvent , Referrable			
Attribute	Datatype	Mul.	Kind	Note
data	VariableDataPrototype	0..1	iref	Data element referenced by event

Table 7.13: DataReceiveErrorEvent

Class	OperationInvokedEvent			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::RTE Events			
Note	The OperationInvokedEvent references the ClientServerOperation invoked by the client.			
Base	ARObject, AbstractEvent , AtpClassifier, AtpFeature, AtpStructure Element, Identifiable , MultilanguageReferrable, RTEEvent , Referrable			
Attribute	Datatype	Mul.	Kind	Note
operation	ClientServerOperation	0..1	iref	The operation to be executed as the consequence of the event.

Table 7.14: OperationInvokedEvent

Class	TimingEvent			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::RTE Events			
Note	TimingEvent references the RunnableEntity that need to be started in response to the TimingEvent			
Base	ARObject, AbstractEvent , AtpClassifier, AtpFeature, AtpStructure Element, Identifiable , MultilanguageReferrable, RTEEvent , Referrable			
Attribute	Datatype	Mul.	Kind	Note
period	TimeValue	1	attr	Period of timing event in seconds. The value of this attribute shall be greater than zero.

Table 7.15: TimingEvent

[constr_2031] Period of [TimingEvent](#) shall be greater than 0 [The value of the attribute period of [TimingEvent](#) shall be greater than 0.]()

Note that it is possible to override the attribute [period](#) on the level of instantiation. See [\[TPS_SWCT_02507\]](#) for more details.

Class	BackgroundEvent			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::RTE Events			
Note	This event is used to trigger RunnableEntities that are supposed to be executed in the background.			
Base	ARObject, AbstractEvent , AtpClassifier, AtpFeature, AtpStructure Element, Identifiable , MultilanguageReferrable, RTEEvent , Referrable			
Attribute	Datatype	Mul.	Kind	Note
–	–	–	–	–

Table 7.16: BackgroundEvent

Class	SwcModeSwitchEvent			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::RTE Events			
Note	This event is raised upon a received mode change.			
Base	ARObject, AbstractEvent , AtpClassifier, AtpFeature, AtpStructure Element, Identifiable , MultilanguageReferrable, RTEEvent , Referrable			
Attribute	Datatype	Mul.	Kind	Note
activation	ModeActivationKind	1	attr	Specifies if the event is activated on entering or exiting the referenced Mode.
mode (ordered)	ModeDeclaration	1..2	iref	Reference to one or two Modes that initiate the SwcModeSwitchEvent.

Table 7.17: SwcModeSwitchEvent

Class	ModeSwitchedAckEvent			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::RTE Events			
Note	The event is raised when the referenced modes have been received or an error occurs.			
Base	ARObject, AbstractEvent , AtpClassifier, AtpFeature, AtpStructure Element, Identifiable , MultilanguageReferrable, RTEEvent , Referrable			
Attribute	Datatype	Mul.	Kind	Note
eventSource	ModeSwitchPoint	1	ref	Mode switch point that triggers the event.

Table 7.18: ModeSwitchedAckEvent

Class	ExternalTriggerOccurredEvent			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::RTE Events			
Note	The event is raised when the referenced trigger have been occurred.			
Base	ARObject, AbstractEvent , AtpClassifier, AtpFeature, AtpStructure Element, Identifiable , MultilanguageReferrable, RTEEvent , Referrable			
Attribute	Datatype	Mul.	Kind	Note
trigger	Trigger	0..1	iref	Reference to the applicable Trigger.

Table 7.19: ExternalTriggerOccurredEvent

Class	InternalTriggerOccurredEvent			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::RTE Events			
Note	The event is raised when the referenced internal trigger have been occurred.			
Base	ARObject, AbstractEvent , AtpClassifier, AtpFeature, AtpStructure Element, Identifiable , MultilanguageReferrable, RTEEvent , Referrable			
Attribute	Datatype	Mul.	Kind	Note
eventSource	InternalTriggeringPoint	1	ref	Internal Triggering Point that triggers the event.

Table 7.20: InternalTriggerOccurredEvent

Class	InitEvent			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::RTE Events			
Note	This RTEEvent is supposed to be used for initialization purposes, i.e. for starting and restarting a partition. It is not guaranteed that all RunnableEntities referenced by this InitEvent are executed before the 'regular' RunnableEntities are executed for the first time. The execution order depends on the task mapping.			
Base	ARObject, AbstractEvent , AtpClassifier, AtpFeature, AtpStructure Element, Identifiable , MultilanguageReferrable, RTEEvent , Referrable			
Attribute	Datatype	Mul.	Kind	Note
—	—	—	—	—

Table 7.21: InitEvent

Class	TransformerHardErrorEvent			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::RTE Events			
Note	The event is raised when data are received which should trigger a Client/Server operation or an external trigger but during transformation of the data a hard transformer error occurred.			
Base	ARObject, AbstractEvent , AtpClassifier, AtpFeature, AtpStructure Element, Identifiable , MultilanguageReferrable, RTEEvent , Referrable			
Attribute	Datatype	Mul.	Kind	Note
operation	ClientServerOperation	0..1	iref	This represents the ClientServerOperation to which the TransformerHardErrorEvent refers to.
trigger	Trigger	0..1	iref	Trigger for which the transformer can trigger this TransformerHardErrorEvent

Table 7.22: TransformerHardErrorEvent

[constr_1397] Existence of attributes of [TransformerHardErrorEvent](#) [For any given [TransformerHardErrorEvent](#), **either** the attribute [TransformerHardErrorEvent.operation](#) **or** [TransformerHardErrorEvent.trigger](#) shall exist.]()

In other words, the attributes [operation](#) and [trigger](#) of meta-class [TransformerHardErrorEvent](#) shall be used mutually exclusive.

[TPS_SWCT_01315] Interaction of RunnableEntity with RTEEvent [As described in the Virtual Functional Bus specification [3], the RunnableEntities of an AtomicSwComponentType can interact with the occurrence of such RTEEvents in two ways:

- the RTE can be instructed to enable a specific RunnableEntity when the RTEEvent occurs
- the RTE can provide WaitPoints, that allow a RunnableEntity to block until an RTEEvent in a set of RTEEvents occurs.

]()

7.3.1 Defining an Event

The description of the SwcInternalBehavior includes a description of all RTEEvents that the SwcInternalBehavior of the AtomicSwComponentType relies on.

[TPS_SWCT_01316] Abstract base class RTEEvent [The meta-class RTEEvent shows up as an “abstract” base-class (see e.g. Figure 7.8) in the meta-model: the exact attributes of the RTEEvent depend on the specific sub-class of RTEEvent that is used for the purpose.]()

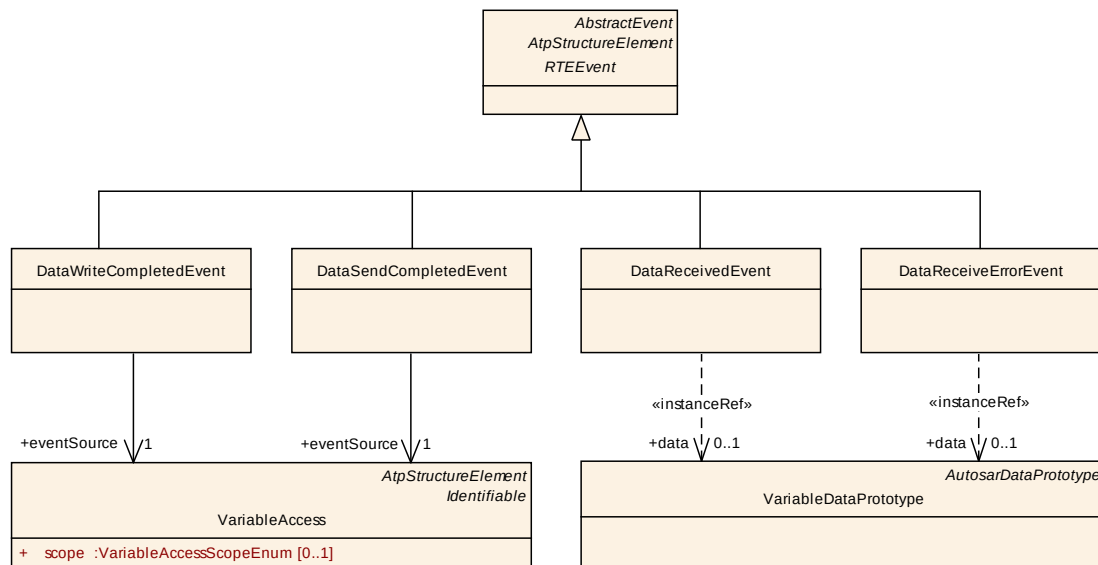


Figure 7.8: RTEEvents used in the context of sender/receiver communication

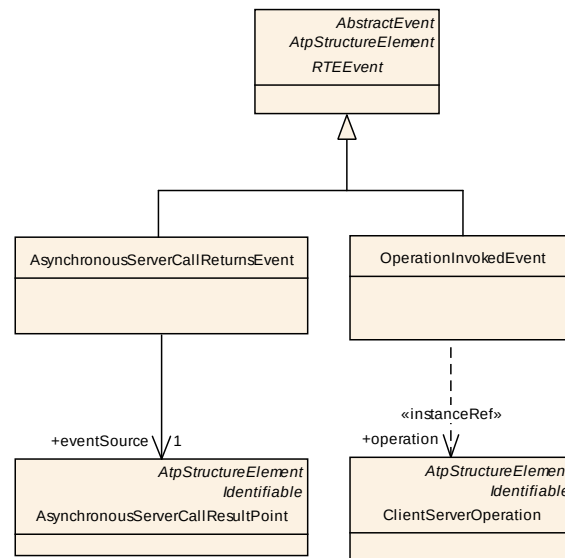


Figure 7.9: RTEEvents used in the context of client/server communication

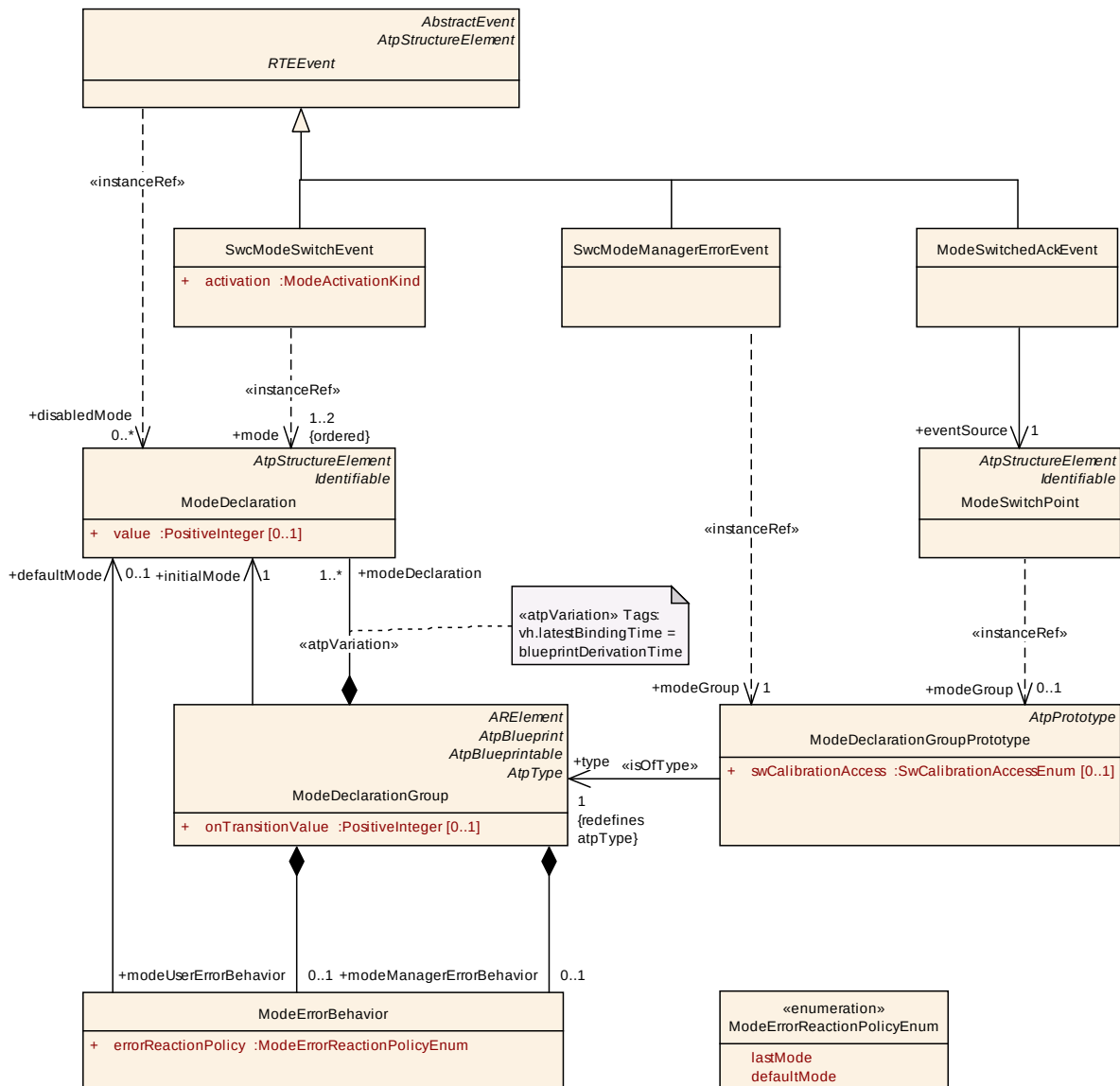


Figure 7.10: RTEEvents used in the context of mode communication

Please note that more explanation about the semantics of the meta-classes `SwcModeManagerErrorEvent` and `ModeErrorBehavior` can be found in section 9.4.

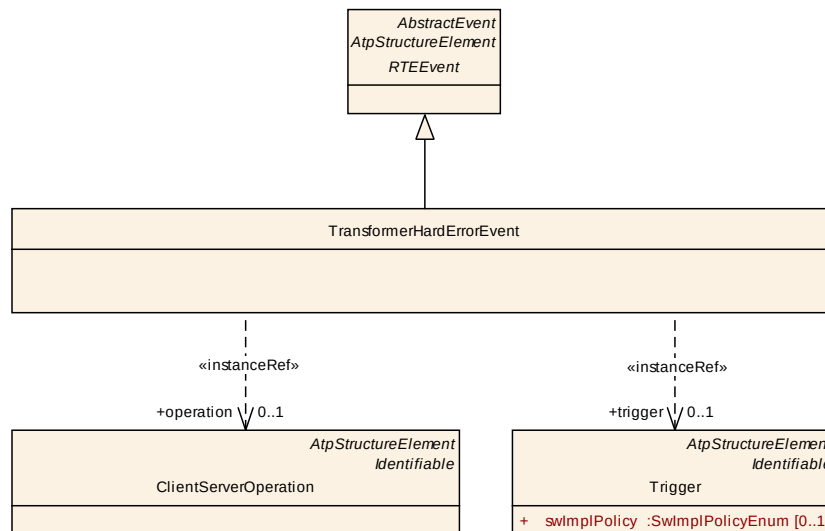


Figure 7.11: **RTEEvent** used in the context of data transformation

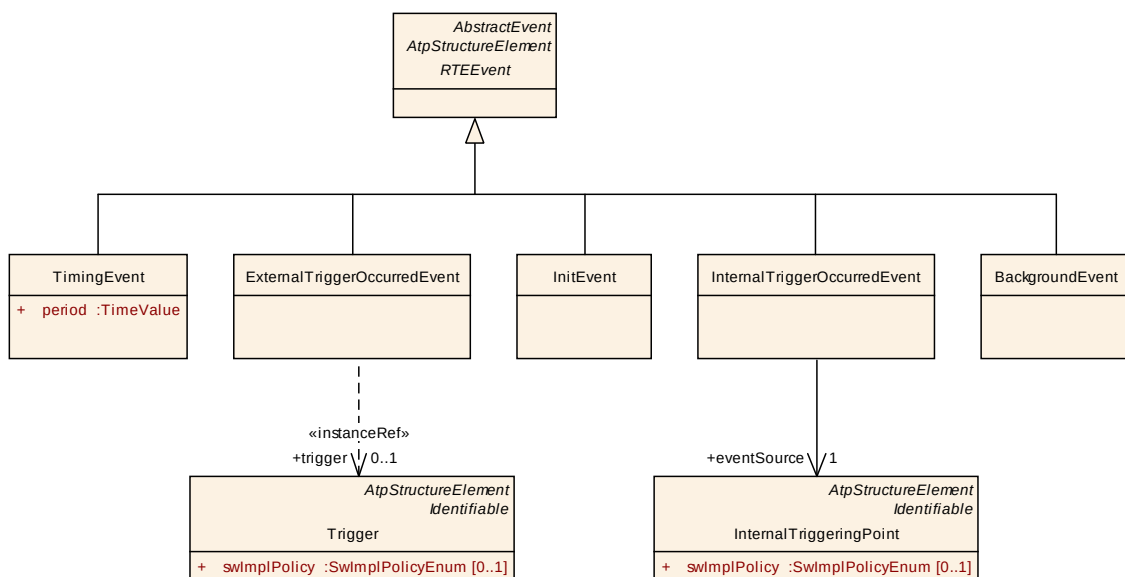


Figure 7.12: **RTEEvents** for purposes other than communication

The details of the various kinds of concrete **RTEEvents** (such as the **TimingEvent**, **DataSendCompletedEvent**, etc.), is described in chapters 7.5.1, 7.5.2 and 7.2.3.

7.3.2 Defining how to Respond to an Event

[TPS_SWCT_01317] RTE triggers **RunnableEntity** in response to occurring **RTEEvent** [If the software-component description contains a reference from an **RTEEvent** to a **RunnableEntity** in the role **startOnEvent** it is the responsibility of the RTE to trigger the execution of the corresponding **RunnableEntity** when the **RTEEvent** occurs.]()

[TPS_SWCT_01318] **RunnableEntity** and **WaitPoint** [In case the **RunnableEntity** wants to block and wait for **RTEEvents** (which makes the **RunnableEntity** into a cat. 2 **RunnableEntity**), the description of the **RunnableEntity** may include the definition of a **WaitPoint**.

Such a **WaitPoint** (see Figure 7.13) contains a reference to an **RTEEvent** that can unblock the specific **WaitPoint**. In other words: the **WaitPoint** will block until the referenced **RTEEvents** occurs or the period specified in the attribute **timeout** expires.]()

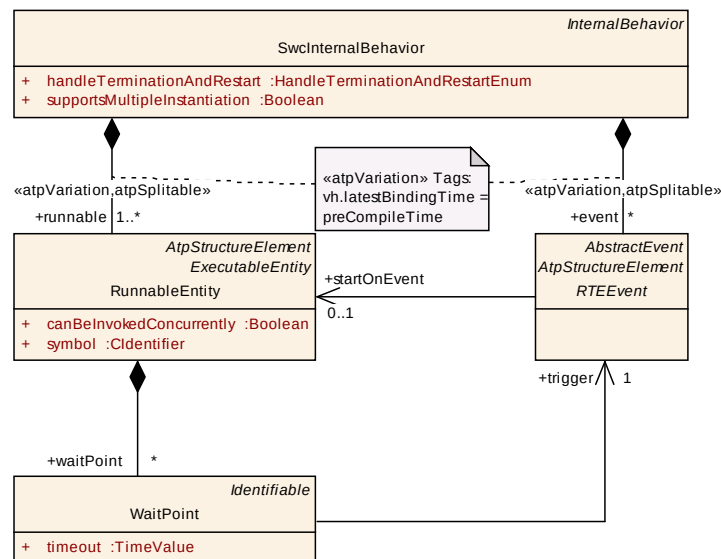


Figure 7.13: Description of the interaction between an **RTEEvent** and **RunnableEntity**s

[constr_1090] **WaitPoint** and **RunnableEntity** [A single **RunnableEntity** can actually wait only at a single **WaitPoint** provided that the **RunnableEntity** can only be scheduled a single time³.]()

[constr_1091] **RTEEvents** that can unblock a **WaitPoint** [The only **RTEEvents** that are qualified for unblocking a **WaitPoint** are:

- **DataReceivedEvent**
- **DataSendCompletedEvent**
- **ModeSwitchedAckEvent**
- **AsynchronousServerCallReturnsEvent**

]()

[TPS_SWCT_01319] **RTEEvent** can be used to trigger **WaitPoints** in different **RunnableEntity**s [It is in general possible that a single **RTEEvent** can be used to trigger **WaitPoints** in different **RunnableEntity**s.]()

³This constraint is valid at least in the OSEK standard where an extended task (that can have wait points) can only exist a single time in the context of the scheduler.

Concerning `DataReceivedEvents` consider as well [\[constr_2021\]](#).

Class	WaitPoint			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::RTE Events			
Note	This defines a wait-point for which the RunnableEntity can wait.			
Base	ARObject, Identifiable , MultilanguageReferrable, Referrable			
Attribute	Datatype	Mul.	Kind	Note
timeout	TimeValue	1	attr	Time in seconds before the WaitPoint times out and the blocking wait call returns with an error indicating the timeout.
trigger	RTEEvent	1	ref	This is the RTEEvent this WaitPoint is waiting for.

Table 7.23: WaitPoint

[constr_1096] SwcModeSwitchEvent and WaitPoint [A [RunnableEntity](#) that has a [WaitPoint](#) shall not be referenced by a [SwcModeSwitchEvent](#).]()

[TPS_SWCT_01320] RunnableEntities of category 2 [[RunnableEntities](#) that aggregate a [WaitPoint](#) are by definition of category 2 and therefore are not required to terminate ever. It is therefore difficult to let a [RunnableEntity](#) of category 2 implement a mode switch.]()

[constr_1097] RunnableEntity that has a WaitPoint [A [RunnableEntity](#) that has a [WaitPoint](#) shall not be referenced by a [RTEEvent](#) that has a reference in the role [disabledMode](#).]()

[TPS_SWCT_01324] Mode switches need to be completed in finite time [Mode switches need to be completed in finite time and a [RunnableEntity](#) that has a [WaitPoint](#) can never guarantee that the [WaitPoint](#) is resolved within finite time.]()

In addition to this, the [RunnableEntity](#) with a [WaitPoint](#) that would be affected by a mode disabling would typically already run when the mode disabling applies. It could not be terminated at this point in time.

7.4 Communication among Runnable Entities

It is taken for granted that particular [RunnableEntities](#) within a specific [AtomicSwComponentType](#) will need to communicate among each other.

[TPS_SWCT_01321] Communication among RunnableEntities [The RTE needs to provide synchronization mechanisms to the [RunnableEntities](#) such that safe (in the multi-threading sense) exchange of data is possible.

In this case, the use of [PortPrototypes](#) is (although technically feasible) not required for the purpose.]([RS_SWCT_00120](#))

[TPS_SWCT_01592] Communication among RunnableEntities of different instances of the same AtomicSwComponentType [The communication among

`RunnableEntity`s of different instances of the same `AtomicSwComponentType` is only supported via `PortPrototypes`. |(RS_SWCT_00120)

Several concepts for implementing communication among `RunnableEntity`s can be identified.

As an introduction, the section 2.3.1 describes the various techniques that the RTE might use to provide efficient interaction between `RunnableEntity`s within one `AtomicSwComponentType`.

Two possible approaches for formal specification of this kind of communication are described:

- Specifying that several `RunnableEntity`s belong in a specific `ExclusiveArea`
- Specifying the data exchanged between the `RunnableEntity`s

7.4.1 Description Possibility 1: Exclusive Area

This section describes how the concept of `ExclusiveAreas` can be used in the description of the `SwcInternalBehavior` of an `AtomicSwComponentType`. Please note that `ExclusiveAreas` are actually owned by the base class of `SwcInternalBehavior`, i.e. `InternalBehavior`. These `ExclusiveAreas` do not imply a specific implementation (e.g. with mutual-exclusion semaphores).

Class	ExclusiveArea			
Package	M2::AUTOSARTemplates::CommonStructure::InternalBehavior			
Note	Prevents an executable entity running in the area from being preempted.			
Base	ARObject,Identifiable,MultilanguageReferrable,Referrable			
Attribute	Datatype	Mul.	Kind	Note
–	–	–	–	–

Table 7.24: ExclusiveArea

[TPS_SWCT_01031] **ExclusiveArea** [An `ExclusiveArea` (please find details about the formal definition of this meta-class in Figure 7.14) merely specifies a constraint on the scheduling policy and configuration of the RTE:

If two or more `RunnableEntity`s refer to the same `ExclusiveArea` only one of these `RunnableEntity`s is allowed to be executed while being inside that `ExclusiveArea`. |(RS_SWCT_00120, RS_SWCT_02090)

In other words: these `RunnableEntity`s shall not run concurrently (preempt each other) while executing inside the `ExclusiveArea`.

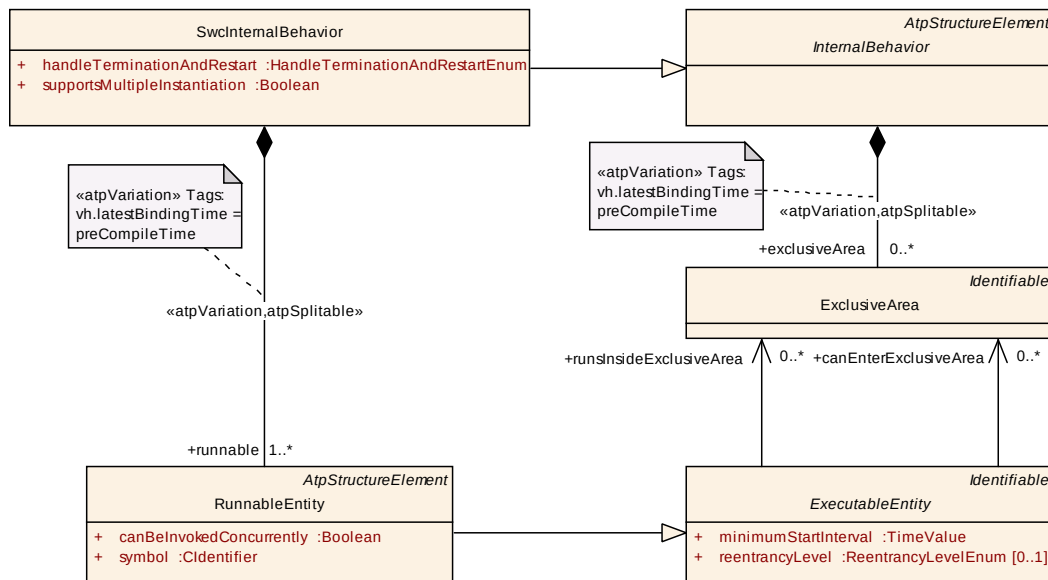


Figure 7.14: Description of logical exclusive areas

[TPS_SWCT_01049] Two ways to use the **ExclusiveAreas** [There are in general two ways to use the **ExclusiveAreas**. During its execution, a **RunnableEntity** can enter and exit an **ExclusiveArea** (in which case **ExecutableEntity.canEnterExclusiveArea** shall exist, see chapter 7.4.1.2).

As an alternative, it can be specified that the entire execution of a given **RunnableEntity** shall be guarded by an **ExclusiveArea** (this requires the existence of **ExecutableEntity.runsInsideExclusiveArea**, see chapter 7.4.1.1).
|(RS_SWCT_00120, RS_SWCT_02090)

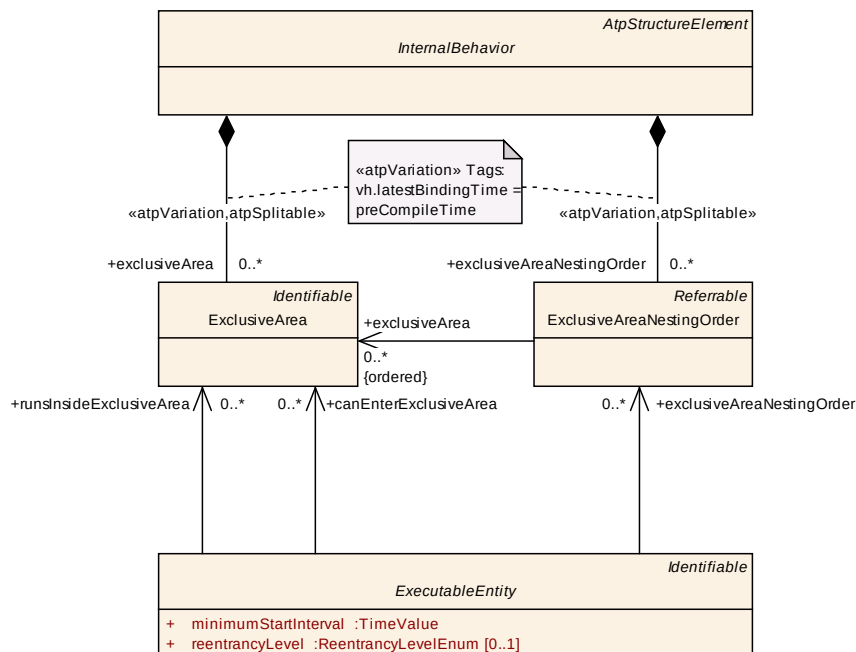


Figure 7.15: Description of nested usage of **ExclusiveArea**

[TPS_SWCT_01457] **ExclusiveAreaNestingOrder** [The optional **ExclusiveAreaNestingOrders** shall (if used at all) describe possible nesting orders (including single **ExclusiveAreas**) which can occur in the **RunnableEntity**. Each possible locking situation requires its own **ExclusiveAreaNestingOrder**.]([RS_SWCT_03055](#))

[TPS_SWCT_01458] Indicate that the locking behavior is fully described for **RunnableEntity** [All **ExclusiveAreas** which are configured in the **InternalBehavior** should be referenced by an **ExclusiveAreaNestingOrder** to indicate that the locking behavior is fully described for this **RunnableEntity**.]([RS_SWCT_03055](#))

[TPS_SWCT_01459] **Locking behavior is not described for this RunnableEntity** [If **ExclusiveAreas** are not referenced by any **ExclusiveAreaNestingOrder** (this is the default scenario), this means that the locking behavior is not described for this **RunnableEntity** and the provided information might be incomplete and cannot be used for a global offline analysis of locking behavior.]([RS_SWCT_03055](#))

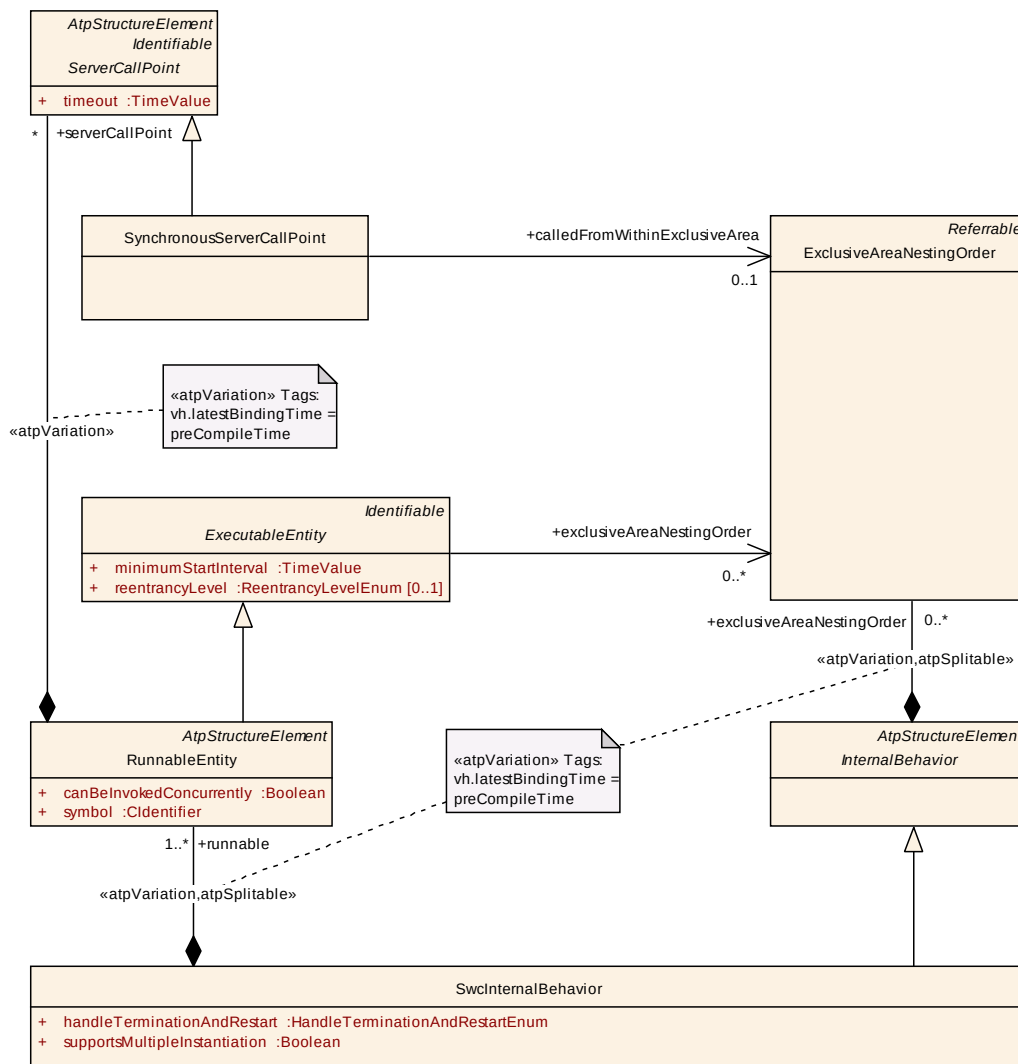


Figure 7.16: Nested usage of **ExclusiveArea and the impact on **SynchronousServerCallPoint****

An [ExclusiveAreaNestingOrder](#) is aggregated by the [InternalBehavior](#) that in turn also owns [RunnableEntity](#).

[TPS_SWCT_01460] Relation of [SynchronousServerCallPoint](#) to [ExclusiveAreaNestingOrder](#) [In case other [RunnableEntity](#)s are invoked synchronously from within the [RunnableEntity](#) the [ExclusiveAreaNestingOrder](#) can then be referenced by one or several [SynchronousServerCallPoints](#) to specify the calling environment of the invoked server with regard to [ExclusiveAreas](#).]([RS_SWCT_03055](#))

The purpose of this configuration is to analyze the resource locking behavior for complete call trees.

Class	ExclusiveAreaNestingOrder			
Package	M2::AUTOSARTemplates::CommonStructure::InternalBehavior			
Note	This meta-class represents the ability to define a nesting order of ExclusiveAreas . A nesting order (that may occur in the executable code) is formally defined to be able to analyze the resource locking behavior.			
Base	ARObject , Referrable			
Attribute	Datatype	Mul.	Kind	Note
exclusi veArea (ordered)	ExclusiveArea	*	ref	This represents a specific scenario of how ExclusiveAreas can be used in terms of the nesting order.

Table 7.25: ExclusiveAreaNestingOrder

7.4.1.1 Entire Runnable Runs in the Exclusive Area

[TPS_SWCT_01050] [RunnableEntity](#) always runs inside an [ExclusiveArea](#) [In the first approach, the formal description specifies that certain [RunnableEntity](#)s always run inside an [ExclusiveArea](#).]([RS_SWCT_00120](#), [RS_SWCT_02090](#))

For example, if the formal description specifies that both [RunnableEntity](#) 'r1' and [RunnableEntity](#) 'r2' run within [ExclusiveArea](#) 's1', the RTE shall make sure that [RunnableEntity](#)s 'r1' and 'r2' never run concurrently; the scheduler should never preempt 'r1' to run 'r2'.

Note that this pattern does not force the RTE to implement this by using semaphores or mutexes that are taken before the [RunnableEntity](#) starts and given when the [RunnableEntity](#) returns. It only obliges the RTE to make sure that both [RunnableEntity](#)s are never running concurrently.

This requirement could be implemented by several of the implementation strategies described above. For example:

1. Scheduling strategy: if, for example, [RunnableEntity](#)s 'r1' and 'r2' are mapped to the same task, the criterion is automatically satisfied. For this purpose it is necessary to make sure that the OS can only execute a single instance of the task into which the [RunnableEntity](#)s are put.

2. Mutual exclusion semaphores: in case 'r1' and 'r2' are mapped to different tasks ('T1', respectively 'T2'), the OS shall make sure that while 'T1' is executing 'r1', 'T2' running 'r2' can never preempt it and vice-versa. This could be implemented by taking a mutual-exclusion semaphore before executing 'r1' (resp. 'r2') in the context of 't1' (resp. 't2') and returning the semaphore on exiting the `RunnableEntity`.

7.4.1.2 Runnable would Dynamically Enter and Leave the Exclusive Area

[TPS_SWCT_01051] `RunnableEntity` explicitly enters and leaves a specific `ExclusiveArea` [In the second approach, the `RunnableEntity` would explicitly make API-calls to the RTE within the implementation of the `RunnableEntity` to enter and leave a specific `ExclusiveArea`.](*RS_SWCT_00120*, *RS_SWCT_02090*)

This could, for example, be implemented by means of the priority ceiling concept described in chapter 2.3.1.3.

Additionally it is possible to define the execution time the `RunnableEntity` will spend in this `ExclusiveArea` segment. Please note that although this aspect is described in [7] the concept can be applied to software-components as well.

7.4.2 Description Possibility 2: Inter-Runnable Variable

For certain cases the `ExclusiveArea` concept does not provide enough information to configure the RTE correctly. In these cases it may be advised to opt for a different approach that is based on the guarded access to variables protected by the RTE.

For the purpose of identifying pieces of data that shall be accessed concurrently from different `RunnableEntity`s formal support is required. In AUTOSAR, this aspect is summarized under the term "inter-runnable variable".

[TPS_SWCT_01052] Inter-runnable variable [These so-called "inter-runnable variables" are described with the element `VariableDataPrototype` aggregated in the role `explicitInterRunnableVariable` or `implicitInterRunnableVariable`.](*RS_SWCT_00120*, *RS_SWCT_02090*)

[TPS_SWCT_01053] Relationship of interchanged data with `RunnableEntity`s [Furthermore, the relationship of these data with `RunnableEntity`s shall be specified.

For this specific purpose, `RunnableEntity` aggregates `VariableAccess` in the roles `readLocalVariable` and `writtenLocalVariable`.

Also, `SwcInternalBehavior` aggregates `VariableDataPrototype` in the roles `explicitInterRunnableVariable` and `implicitInterRunnableVariable`.

The connection between `RunnableEntity` and the `explicitInterRunnableVariable` and `implicitInterRunnableVariable` is created if the reference `Au-`

`tosarVariableRef.localVariable` to the respective `VariableDataPrototype` exists.](*RS_SWCT_00120, RS_SWCT_02090*)

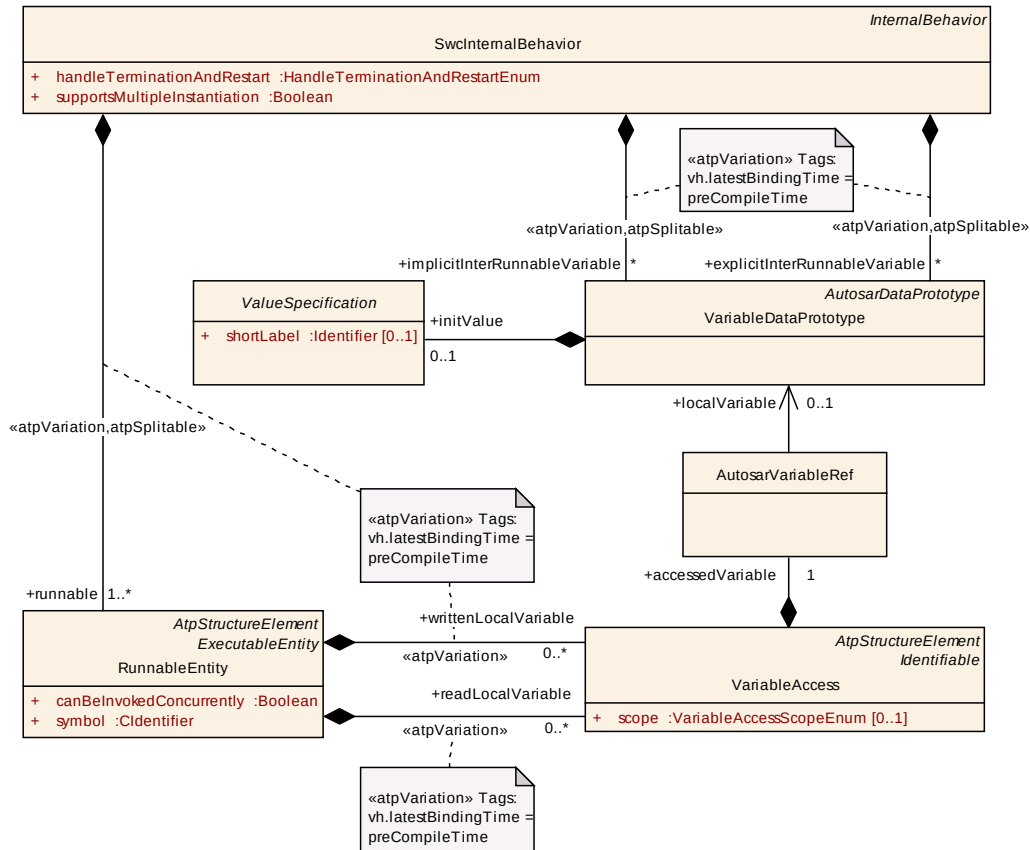


Figure 7.17: `implicitInterRunnableVariable` vs. `explicitInterRunnableVariable`

[TPS_SWCT_01521] Use `AutosarVariableRef.localVariable` for referencing inter-runnable variables [A `RunnableEntity` that defines a `VariableAccess` in role `writtenLocalVariable` and `readLocalVariable` shall make use of `AutosarVariableRef.localVariable`.]()

[constr_2026] Referenced `VariableDataPrototype` from `AutosarVariableRef` of `VariableAccess` in role `writtenLocalVariable` and `readLocalVariable` [A `VariableDataPrototype` in the `localVariable` reference needs to be owned by the same `SwcInternalBehavior` as this `RunnableEntity` belongs to, and the referenced `VariableDataPrototype` has to be defined in the role `implicitInterRunnableVariable` or `explicitInterRunnableVariable`.]()

Obviously, the data type of an `implicitInterRunnableVariable` or `explicitInterRunnableVariable` is described by the data type of the `VariableDataPrototype` (which is derived from `DataPrototype`).

[TPS_SWCT_01637] Initial value for a specific `implicitInterRunnableVariable` or `explicitInterRunnableVariable` [It is possible (but not mandatory)

to define an initial value for a specific `implicitInterRunnableVariable` or `explicitInterRunnableVariable`.

For this purpose the `VariableDataPrototype` in the role of `explicitInterRunnableVariable` or `implicitInterRunnableVariable` is able to aggregate a `ValueSpecification` in the role `initValue`. [\]\(RS_SWCT_02090\)](#)

The statement made by [\[TPS_SWCT_01637\]](#) is reflected by Figure 7.17

[TPS_SWCT_01522] No initial value is specified for `implicitInterRunnableVariable` or `explicitInterRunnableVariable` [\[](#) Please note that the behavior is undefined if no initial value is specified and a `RunnableEntity` reads an `implicitInterRunnableVariable` or `explicitInterRunnableVariable` before it is actually written to by another `RunnableEntity`. [\]\(](#)

As already mentioned before, the concept of an "inter-runnable variable" can be used in *two different flavors* This is indicated by the two different roles `explicitInterRunnableVariable` or `implicitInterRunnableVariable` in which the `VariableDataPrototype` serving as the "inter-runnable variable" is aggregated.

These resemble the communication principles applied for the communication on the level of `SwComponentTypes`.

Please note that the two different kinds of inter-runnable variables are accessed via different RTE [2] API calls.

[TPS_SWCT_01054] Semantics of the `explicitInterRunnableVariable` [\[](#) The semantics of the `explicitInterRunnableVariable` is that *explicit* implies the direct access to the value of an `VariableDataPrototype` used in the role `explicitInterRunnableVariable` or `implicitInterRunnableVariable`.

By this means it is possible to get different values for a specific `VariableDataPrototype` each time the corresponding API call is executed. [\]\(RS_SWCT_00120, RS_SWCT_02090\)](#)

[TPS_SWCT_01055] Semantics of `implicitInterRunnableVariable` [\[](#) The `implicitInterRunnableVariable` corresponds to an execution model where the value of an `VariableDataPrototype` does not change (for the reading `RunnableEntity`, obviously) during the runtime of a `RunnableEntity`.

This approach is in detail described in chapter 2.3.1.4. [\]\(RS_SWCT_00120, RS_SWCT_02090\)](#)

[constr_1296] `DataPrototypes` used as `explicitInterRunnableVariable` or `implicitInterRunnableVariable` and category `DATA_REFERENCE` [\[](#) A `VariableDataPrototype` shall not be aggregated by `SwcInternalBehavior` in either the role `explicitInterRunnableVariable` or `implicitInterRunnableVariable` if the `VariableDataPrototype` (after potential indirections via `TYPE_REFERENCE` are resolved) is either typed by or mapped to an `ImplementationDataType` of category `DATA_REFERENCE`. [\]\(](#)

7.4.3 Inter Runnable Triggering

The concept of *inter-runnable triggering* allows one [RunnableEntity](#) to trigger another [RunnableEntity](#) within an [AtomicSwComponentType](#). This approach conceptually supports the decoupling of calculation and processing sequences inside a software-component.

By mappings of the [InternalTriggerOccurredEvents](#) to OS Tasks running at different priorities the triggered [RunnableEntities](#) are in turn executed with a different priority as the triggering [RunnableEntity](#).

For example, a cyclically triggered [RunnableEntity](#) which shall not exceed a certain worst case execution time (WCET) activates a second [RunnableEntity](#) if an error occurred in order to be able to execute a (potentially) time-consuming exception-handling on a lower level of priority.

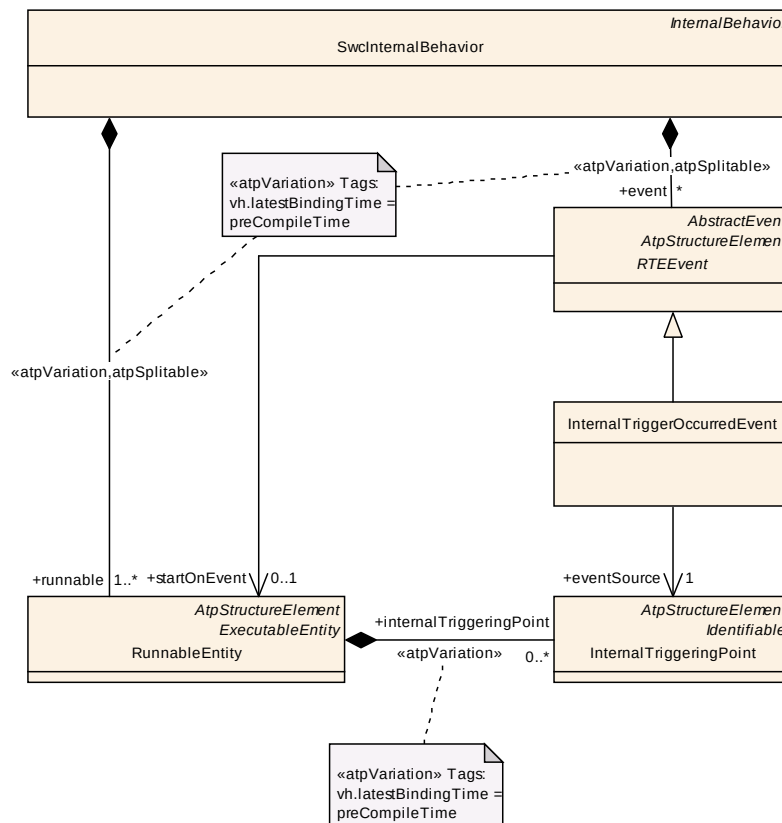


Figure 7.18: Model of software-component Inter Runnable Triggering

As illustrated in Figure 7.18 the triggering [RunnableEntity](#) needs an [InternalTriggeringPoint](#).

The activation of [RunnableEntities](#) in the same software-component instance is affected through the generic event-handling mechanism.

[TPS_SWCT_01523] Internal trigger event [A [RunnableEntity](#) that shall be activated at the occurrence of an internal trigger event is defined by means of an [In-](#)

`ternalTriggerOccurredEvent` which references the particular `InternalTriggeringPoint` and additionally the to-be-activated `RunnableEntity`. `]()`

[TPS_SWCT_01022] Queued processing of internal trigger [The attribute `InternalTriggeringPoint.swImplPolicy` can be used to specify a requirement whether or not the internal triggering of the enclosing `RunnableEntity` using the given `InternalTriggeringPoint` shall be queued. `]()`

[constr_1182] Allowed values for `InternalTriggeringPoint.swImplPolicy` [The **only** allowed values for the attribute `swImplPolicy` of meta-class `InternalTriggeringPoint` are either `STANDARD` (in which case the processing of the internal triggering does not use a queue) or `QUEUED` (in which case the processing of internal triggering positively uses a queue). `]()`

Class	InternalTriggeringPoint			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::Trigger			
Note	If a <code>RunnableEntity</code> owns a <code>InternalTriggeringPoint</code> it is entitled to trigger the execution of <code>RunnableEntities</code> of the corresponding software-component.			
Base	ARObject, AtpClassifier, AtpFeature, AtpStructureElement, Identifiable , Multilanguage Referrable, Referrable			
Attribute	Datatype	Mul.	Kind	Note
swImplPolicy	SwImplPolicyEnum	0..1	attr	This attribute, when set to value <code>queued</code> , allows for a queued processing of Triggers.

Table 7.26: InternalTriggeringPoint

Class	InternalTriggerOccurredEvent			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::RTE Events			
Note	The event is raised when the referenced internal trigger have been occurred.			
Base	ARObject, AbstractEvent , AtpClassifier, AtpFeature, AtpStructureElement, Identifiable , Multilanguage Referrable, RTEEvent , Referrable			
Attribute	Datatype	Mul.	Kind	Note
eventSource	InternalTriggeringPoint	1	ref	Internal Triggering Point that triggers the event.

Table 7.27: InternalTriggerOccurredEvent

The description of the corresponding *external* trigger communication is contained in chapter 7.5.3.

7.5 Data Access of RunnableEntities

This section describes the communication properties of an `AtomicSwComponentType`. This is done mainly from the point of view of a `RunnableEntity` (the concept of a `RunnableEntity` is introduced in chapter 7.2).

However, the usage of a [PortPrototype](#) in a specific role within an [AtomicSwComponentType](#) also has an impact on communication behavior.

7.5.1 RunnableEntities and Sender Receiver Communication

This section describes aspects relevant for the sender-receiver communication of a software-component. These mainly influence the behavior and API of the AUTOSAR RTE.

[TPS_SWCT_01322] Interaction patterns for the application of the sender-receiver paradigm [The possible interaction patterns for the application of the sender-receiver paradigm are explained, namely:

1. Data-access in a cat. 1 [RunnableEntity](#),
2. explicit sending,
3. the [DataSendCompletedEvent](#): dealing with the success/failure of an explicit send, and
4. the [DataReceivedEvent](#): responding to the reception of data
5. the [DataReceiveErrorEvent](#): notifying an error concerning the reception of data.

]([RS_SWCT_00200](#))

7.5.1.1 Terminology

The AUTOSAR meta-model foresees two different approaches for sender-receiver communication. These are described in detail in chapters [7.5.1.2](#) and [7.5.1.3](#). However, it turned out that it is rather cumbersome to discuss issues of communication approaches directly on the basis of meta-classes and their attributes.

Therefore, it seems appropriate to introduce a dedicated terminology for this purpose. The approach eventually selected was originally introduced by the contributors to the RTE specification.

This terminology proposes to use the term “implicit” for communication based on *data-access* (for more information about details of this approach please consult chapter [7.5.1.2](#)) and “explicit” for communication based on so-called *data-points* (please refer to chapter [7.5.1.3](#)).

The motivation for the differentiation between “implicit” and “explicit” was originally the characteristics of the RTE specification that foresaw an API for handling a [dataSendPoint](#) or [dataReceivePointByValue](#) in contrast to the *data-access* that was supposed to be part of the function signature (therefore, no API was required) of a specific [RunnableEntity](#).

Although the specification of the RTE changed in the meantime (and the original motivation no longer applies) it turned out that the terminology based on “implicit” and “explicit” communication” was already widely used within AUTOSAR.

As no consensus could be reached over alternative proposals this terminology approach is taken over by this document as well.

7.5.1.2 Data Access

[TPS_SWCT_01323] Read and write access to a `dataElement` [The `SwcInternalBehavior` may specify that a `RunnableEntity` needs read-access (respectively write-access) to the `VariableDataPrototypes` in the role `dataElement` of an `RPortPrototype` (respectively `PPortPrototype`, or `PRPortPrototype`).]([RS_SWCT_00200](#))

[TPS_SWCT_01325] Read and write access is only applicable for `RunnableEntities` of category 1 [The usage of the data-access mechanism to the `VariableDataPrototypes` is appropriate for cat. 1 `RunnableEntities` only because it by concept guarantees finite response time (as opposed to e.g. unlimited blocking wait for some data).]([RS_SWCT_00200](#))

For more explanation, let's suppose a cat. 2 `RunnableEntity` would have a `dataReadAccess` and a `dataWriteAccess`. The received `dataElement` would be updated **before** the `RunnableEntity` actually starts being executed and even if the `RunnableEntity` runs for a very long time the value of the `dataElement` would remain as is and never change.

On the other hand, the `RunnableEntity` might use its `dataWriteAccess` to perform a write access on the `dataElement` but the actual value might never make it beyond the `RunnableEntity` because

1. the latter is not required to terminate ever and
2. the actual write access is executed *after* the `RunnableEntity` terminates.

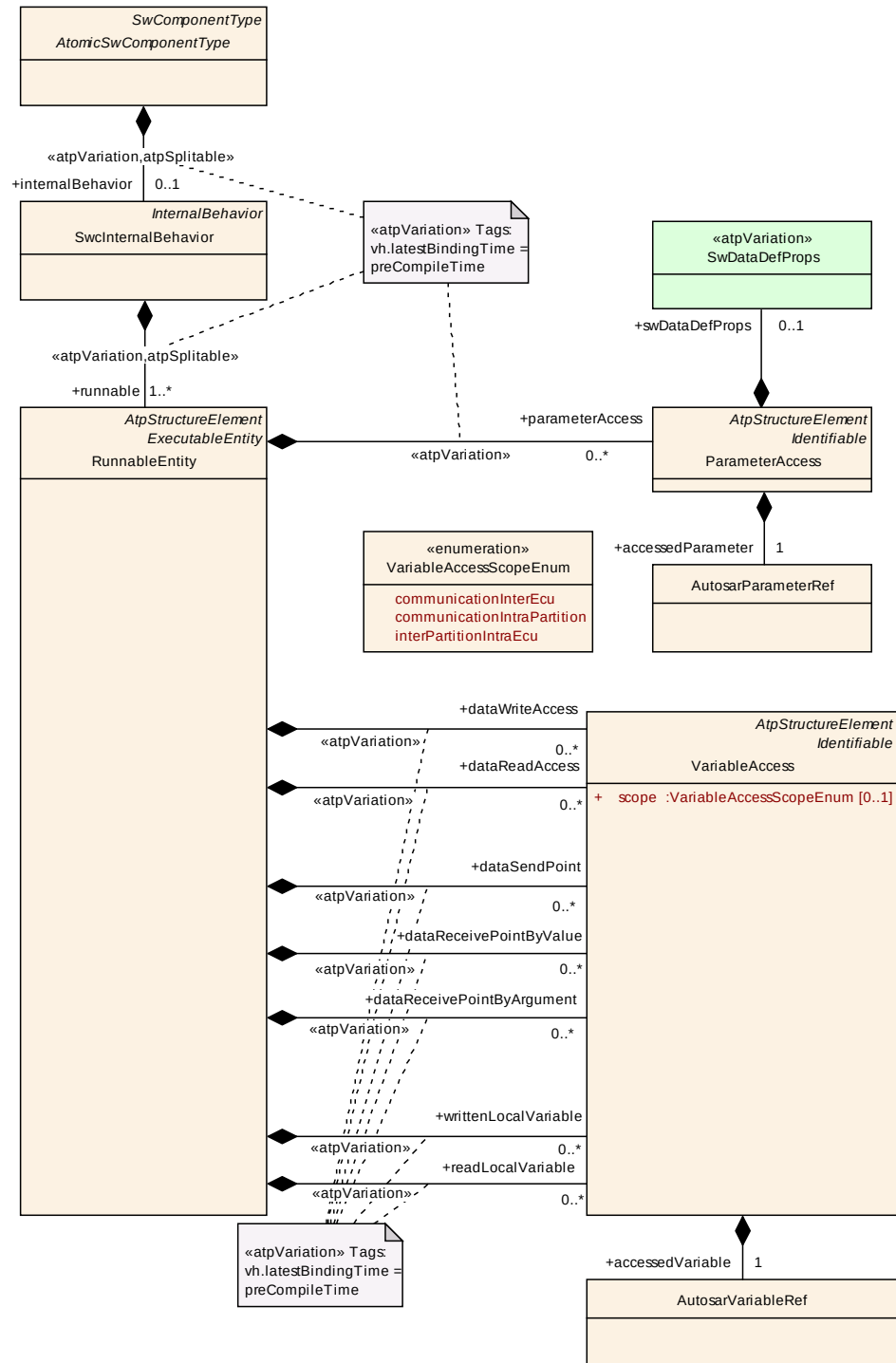


Figure 7.19: DataReadAccess and DataWriteAccess

Class	VariableAccess			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::Data Elements			
Note	The presence of a VariableAccess implies that a RunnableEntity needs access to a VariableDataPrototype. The kind of access is specified by the role in which the class is used.			
Base	ARObject, AtpClassifier, AtpFeature, AtpStructureElement, Identifiable , Multilanguage Referrable, Referrable			
Attribute	Datatype	Mul.	Kind	Note
accessedVariable	AutosarVariableRef	1	aggr	This denotes the accessed variable.
scope	VariableAccessScopeEnum	0..1	attr	This attribute allows for constraining the scope of the corresponding communication. For example, it possible to express whether the communication is intended to cross the boundary of an ECU or whether it is intended not to cross the boundary of a single partition.

Table 7.28: VariableAccess

Enumeration	VariableAccessScopeEnum
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::Data Elements
Note	This enumeration defines scopes for communication.
Literal	Description
communicationInterEcu	This case is foreseen to express that the corresponding communication shall be considered inter-ECU, i.e. it will cross the ECU boundary. This is considered the default case.
communicationIntraPartition	This case is foreseen to express that the corresponding communication shall not cross the boundary of a partition.
interPartitionIntraEcu	In this case the communication shall cross the boundaries of partitions within one ECU but it shall not cross the boundaries of the ECU itself.

Table 7.29: VariableAccessScopeEnum

[TPS_SWCT_01326] Constrain the scope of a specific communication [The purpose of the attribute [scope](#) of meta-class [VariableAccess](#) is to constrain the scope of the corresponding communication.

The main use-case for this ability is the development of a software-component where certain end-points of communication from or to the software-component are known to fulfill a certain constraint, e.g. execute within the same partition.]([RS_SWCT_00200](#))

[TPS_SWCT_01328] Default value of attribute [scope](#) [The default value of attribute [scope](#) is set to [communicationInterEcu](#).]([RS_SWCT_00200](#))

[constr_1141] Applicability of the [scope](#) attribute [

The attribute [scope](#) of meta-class [VariableAccess](#) shall **only** be applied with respect to the aggregation of [VariableAccess](#) in the following roles:

- `dataReadAccess`
- `dataWriteAccess`
- `dataSendPoint`
- `dataReceivePointByValue`
- `dataReceivePointByArgument`

⌋()

[TPS_SWCT_01329] Access to specific data is implemented by means of aggregating the meta-class `VariableAccess` in specific roles ⌈ Please note that from the formal point of view access to specific data is implemented by means of aggregating the meta-class `VariableAccess` in specific roles.

This means that `dataReadAccess` for a read-access while the write-access is defined by means of aggregating `VariableAccess` in the role `dataWriteAccess`.
⌋(*RS_SWCT_00200*)

This aspect is depicted in Figure 7.19.

The following constraints apply to the reference target of the `AutosarVariableRef` of `VariableAccess` in role `dataReadAccess` or `dataWriteAccess`.

[constr_2002] Referenced `VariableDataPrototype` from `AutosarVariableRef` of `VariableAccess` in role `dataReadAccess` ⌈ A `VariableAccess` in the role `dataReadAccess` shall refer to an `RPortPrototype` or `PRPortPrototype` that is typed by either a `SenderReceiverInterface` or a `NvDataInterface`. ⌋()

[constr_2003] Referenced `VariableDataPrototype` from `AutosarVariableRef` of `VariableAccess` in role `dataWriteAccess` ⌈ A `VariableAccess` in the role `dataWriteAccess` shall refer to a `PPortPrototype` or `PRPortPrototype` that is typed by either a `SenderReceiverInterface` or a `NvDataInterface`. ⌋()

By access with `VariableAccess` in the `dataReadAccess` role always the last value of the `VariableDataPrototype` buffered before the `RunnableEntity` starts will be read during the execution of the `RunnableEntity`.

It would therefore not make any sense to provide a queue of values for the purpose of accessing a `dataElement` in the role `dataReadAccess`.

[constr_2020] `dataReadAccess` can not be used for queued communication ⌈ The `swImplPolicy` of the `VariableDataPrototype` referenced by a `VariableAccess` in role `dataReadAccess` shall **not** be set to `queued`. ⌋()

[constr_1256] Acknowledgement feedback in n:1 writer case ⌈ Within the scope of one `SwcInternalBehavior`, it is **not** allowed that two or more aggregated `RunnableEntity`s own either `dataSendPoints` or `dataWriteAccesses` that in turn point to the identical `accessedVariable.autosarVariable.targetDataPrototype` **if** the attribute `transmissionAcknowledge` exists in the context of the `SenderComSpec` owned by the `dataSendPoint.accessedVariable.autosar-`

`Variable.portPrototype` (or the respective construct for `dataWriteAccess`) that also refers to said `dataElement`. `]()`

The background of [\[constr_1256\]](#) is that if two or more `RunnableEntity`s exist that can write to the identical `dataElement` it may happen that more than one `RunnableEntity` actually write to the respective `dataElement` **before** the “first” acknowledgement is received. In this case it will never be possible to determine exactly which transmission has been acknowledged.

7.5.1.3 Explicit Sending and Receiving

[TPS_SWCT_01330] `RunnableEntity` can also have `dataSendPoints` `[` A `RunnableEntity` can also have `dataSendPoints` (i.e. aggregate `VariableAccess` in the role `dataSendPoint`). `]`

Using an `instanceRef` association, these eventually reference a `VariableDataPrototype` in the context of an `AbstractProvidedPortPrototype`, owned by the `AtomicSwComponentType` that is associated with the `RunnableEntity` that in turn owns the `dataSendPoint`. `](RS_SWCT_00200)`

[constr_2004] Referenced `VariableDataPrototype` from `AutosarVariableRef` of `VariableAccess` in role `dataSendPoint` `[` A `VariableAccess` in the role `dataSendPoint` shall refer to a `PPortPrototype` or `PRPortPrototype` that is typed by either a `SenderReceiverInterface` or a `NvDataInterface`. `]`

[TPS_SWCT_01331] `dataWriteAccess` vs. `dataSendPoint` `[` As opposed to the `dataWriteAccess`:

- Using the `dataSendPoint`, the `RunnableEntity` needs to explicitly “send” through an API; when using a `dataWriteAccess`, the `RunnableEntity` only needs to modify the value of certain variables.
- Using `dataSendPoint`, the `Runnable` can decide to “send” an arbitrary number of times; when using `dataWriteAccess` the new value of the `VariableDataPrototype` is not made available before the `RunnableEntity` returns (exits the “Running” state).
- The presence of a `dataSendPoint` per definition lets the corresponding `RunnableEntity` attain cat. 1B.

`](RS_SWCT_00200)`

For more details, please refer to section [4.9](#).

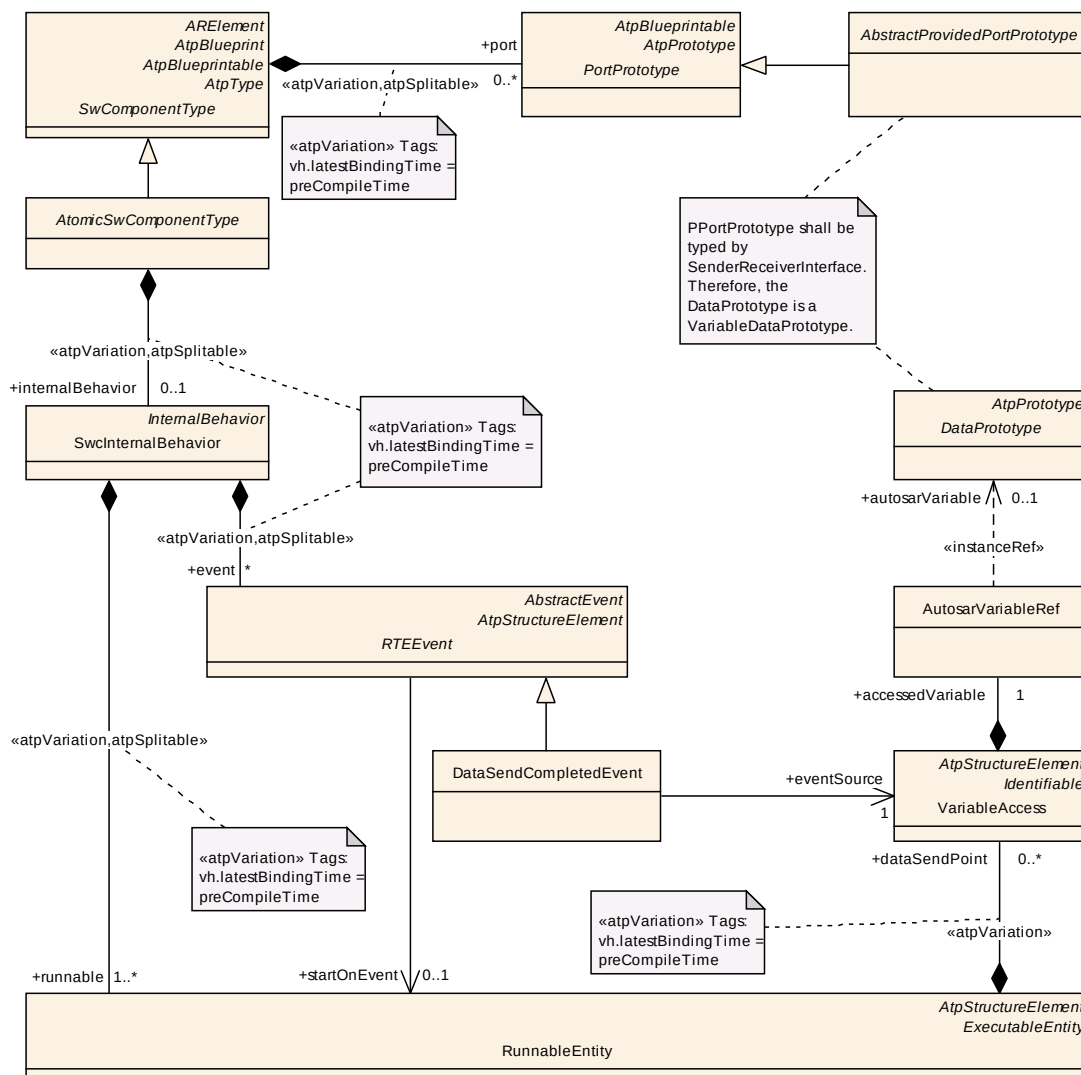


Figure 7.20: DataSendPoint

[TPS_SWCT_01332] `dataReceivePointByValue` vs. `dataReceivePointByArgument` | In analogy to explicitly sending data it is also possible to define explicit polling for new available data through a `dataReceivePointByValue` or `dataReceivePointByArgument` as shown in Figure 7.21. | ()

[constr_1277] **SwDataDefProps.swImplPolicy** of a **VariableDataPrototype** referenced by a **VariableAccess** aggregated in the role **dataReceivePointByValue** **┐** The **SwDataDefProps.swImplPolicy** of a **VariableDataPrototype** referenced by a **VariableAccess** aggregated in the role **dataReceivePointByValue** shall not be set to **queued**. **|()**

Rationale for [constr_1277]: when using the return value of the applicable RTE API function to return the value of a `VariableDataPrototype` there is no way⁴ to provide an indication that the queue is empty. Therefore, the only safe approach is to not permit this scenario at all, hence the constraint.

⁴That is, other than to use a function argument to return the status of the queue but that would obviously beat the purpose of the API function.

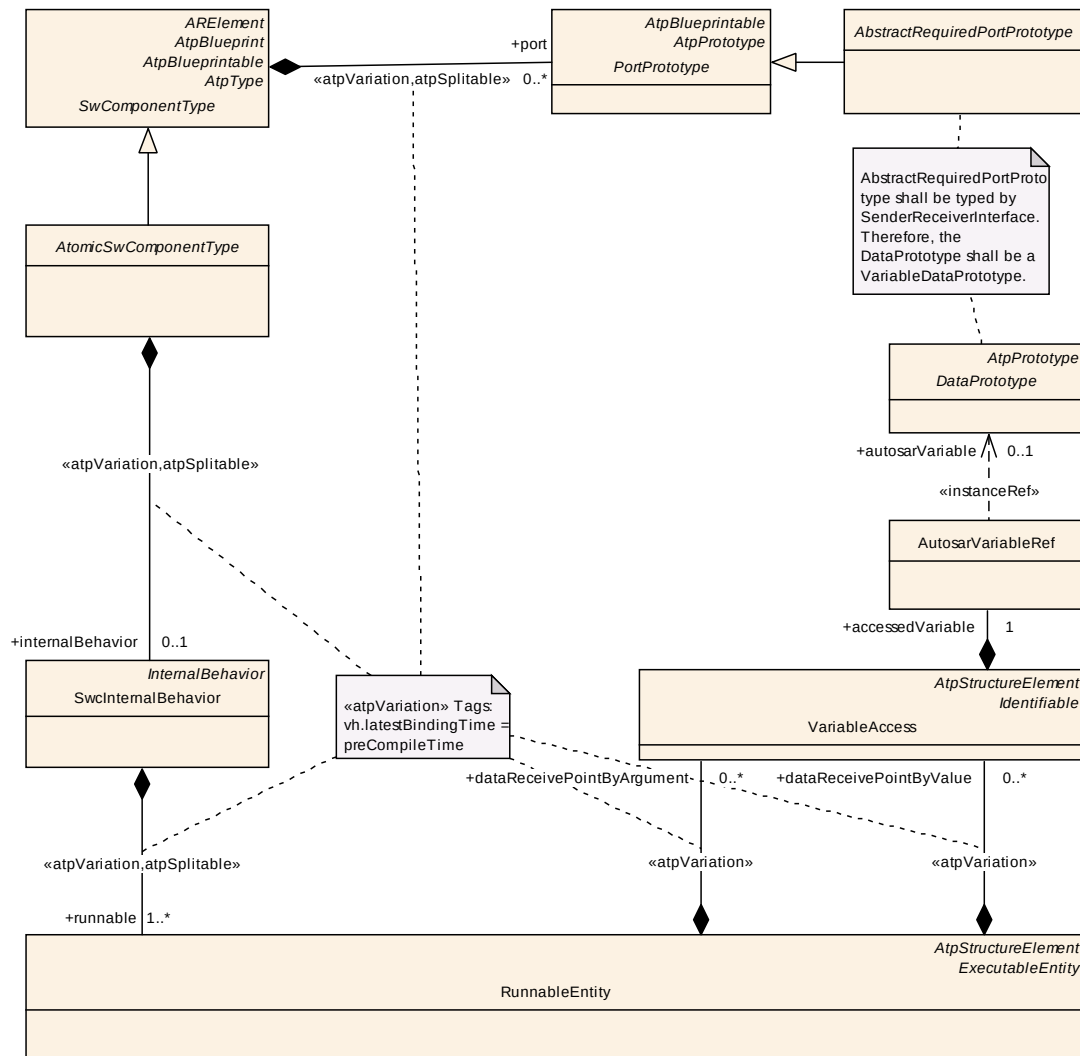


Figure 7.21: Definition of an explicit request to receive data

[TPS_SWCT_01333] **dataReceivePointByValue/dataReceivePointByArgument vs. dataReadAccess** | By using a `dataReceivePointByValue` or `dataReceivePointByArgument` instead of `dataReadAccess` the constraining access to the referenced `VariableDataPrototype` (other `RunnableEntity`s shall not change the `VariableDataPrototype` during the read execution) is limited to a short, well-defined amount of time. | (*RS SWCT 00200*)

[TPS_SWCT_01334] RunnableEntitys of category 1 may have dataReceivePointByValues/dataReceivePointByArguments [Therefore, category 1 RunnableEntitys may also have dataReceivePointByValues/dataReceivePointByArguments and consequently become RunnableEntitys of category 1B, see section 7.2.4.4. | (RS SWCT 00200)

Similar to the `dataReadAccess`, constraints apply to the reference target of the `AutosarVariableRef` of `VariableAccess` in role `dataReceivePointByValue` or `dataReceivePointByArgument`.

[constr_2005] Referenced `VariableDataPrototype` from `AutosarVariableRef` of `VariableAccess` in role `dataReceivePointByValue` or `dataReceivePointByArgument` [A `VariableAccess` in the role `dataReceivePointByValue` or `dataReceivePointByArgument` shall refer to an `RPortPrototype` or `PRPortPrototype` that is typed by either a `SenderReceiverInterface` or an `NvDataInterface`.]()

[TPS_SWCT_01335] Combine `dataReceivePointByValue` or `dataReceivePointByArgument` with a `WaitPoint` [In general, it is possible to combine a `dataReceivePointByValue` or `dataReceivePointByArgument` with a `WaitPoint` in the scope of a particular `RunnableEntity`.

This allows for a call to a blocking receive routine implemented by the RTE. The `timeout` attribute of meta-class `WaitPoint` can be used to specify the time until the blocking call expires.

But in case of non-queued communication it is **not supported** that a `DataReceivedEvent` is used in combination with a `WaitPoint` (see [constr_2021]). This contradicts the approach of the last-is-best semantics.]([RS_SWCT_00200](#))

[constr_2021] `WaitPoint` referencing a `DataReceivedEvent` can not be used for non-queued communication [A `WaitPoint` referencing a `DataReceivedEvent` is permitted **if and only if** the `swImplPolicy` of the `VariableDataPrototype` referenced by this `DataReceivedEvent` is set to `queued`.]()

Please note however, that in this case (in response to the presence of a `WaitPoint`) the `RunnableEntity` becomes category 2.

7.5.1.4 DataSendCompletedEvent

[TPS_SWCT_01336] `dataSendPoint` also allows for the definition of a `DataSendCompletedEvent` [The `dataSendPoint` also allows for the definition of a `DataSendCompletedEvent`, as shown in Figure 7.20. This `RTEEvent` occurs when the data has been successfully sent or when an error has occurred during sending.]([RS_SWCT_00200](#))

Please note that this feature can only be used if the `AtomicSwComponentType` describes the meaning of success or failure of the send operation.

In particular, via a `SenderComSpec` class different acknowledgement requests (in this case: successful transmission) can be attached to a `PPortPrototype` or `PRPortPrototype`, as is shown in Figure 4.33.

This will configure the RTE such that when data is sent the RTE will try to obtain the specified acknowledgement; possibly by waiting a certain `timeout` period.

Class	DataSendCompletedEvent			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::RTE Events			
Note	The event is raised when the referenced data elements have been sent or an error occurs.			
Base	ARObject, AbstractEvent , AtpClassifier, AtpFeature, AtpStructure Element, Identifiable , MultilanguageReferrable, RTEEvent , Referrable			
Attribute	Datatype	Mul.	Kind	Note
eventSource	VariableAccess	1	ref	The variable access that triggers the event.

Table 7.30: DataSendCompletedEvent

[constr_2033] Timeout of [DataSendCompletedEvent](#) [The `timeout` value of a [WaitPoint](#) associated with a [DataSendCompletedEvent](#) shall have the same value as the corresponding value of [TransmissionAcknowledgementRequest.timeout](#).]()

7.5.1.5 DataWriteCompletedEvent

[TPS_SWCT_01557] [dataWriteAccess](#) also allows for the definition of a [DataWriteCompletedEvent](#) [The `dataWriteAccess` also allows for the definition of a [DataWriteCompletedEvent](#), as shown in Figure 7.22. This [RTEEvent](#) occurs when the data has been successfully sent or when an error has occurred during sending.]([RS_SWCT_00200](#))

Please note that this feature can only be used if the [AtomicSwComponentType](#) describes the meaning of success or failure of the send operation.

In particular, via a [SenderComSpec](#) class different acknowledgement requests (in this case: successful transmission) can be attached to a [PPortPrototype](#) or [PRPortPrototype](#), as is shown in Figure 4.33.

[TPS_SWCT_01558] [DataWriteCompletedEvent](#) cannot be combined with a [WaitPoint](#) [Please note that a [DataWriteCompletedEvent](#) cannot be associated with a [WaitPoint](#), see [\[constr_1091\]](#).]([RS_SWCT_00200](#))

However, it is possible to configure the RTE such that when data is sent, the RTE will try to obtain the specified acknowledgement; possibly by waiting a certain `timeout` period.

Class	DataWriteCompletedEvent			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::RTE Events			
Note	This event is raised if an implicit write access was successful or an error occurred.			
Base	ARObject, AbstractEvent , AtpClassifier, AtpFeature, AtpStructure Element, Identifiable , MultilanguageReferrable, RTEEvent , Referrable			
Attribute	Datatype	Mul.	Kind	Note

Attribute	Datatype	Mul.	Kind	Note
eventSource	VariableAccess	1	ref	The variable access that triggers the event.

Table 7.31: DataWriteCompletedEvent

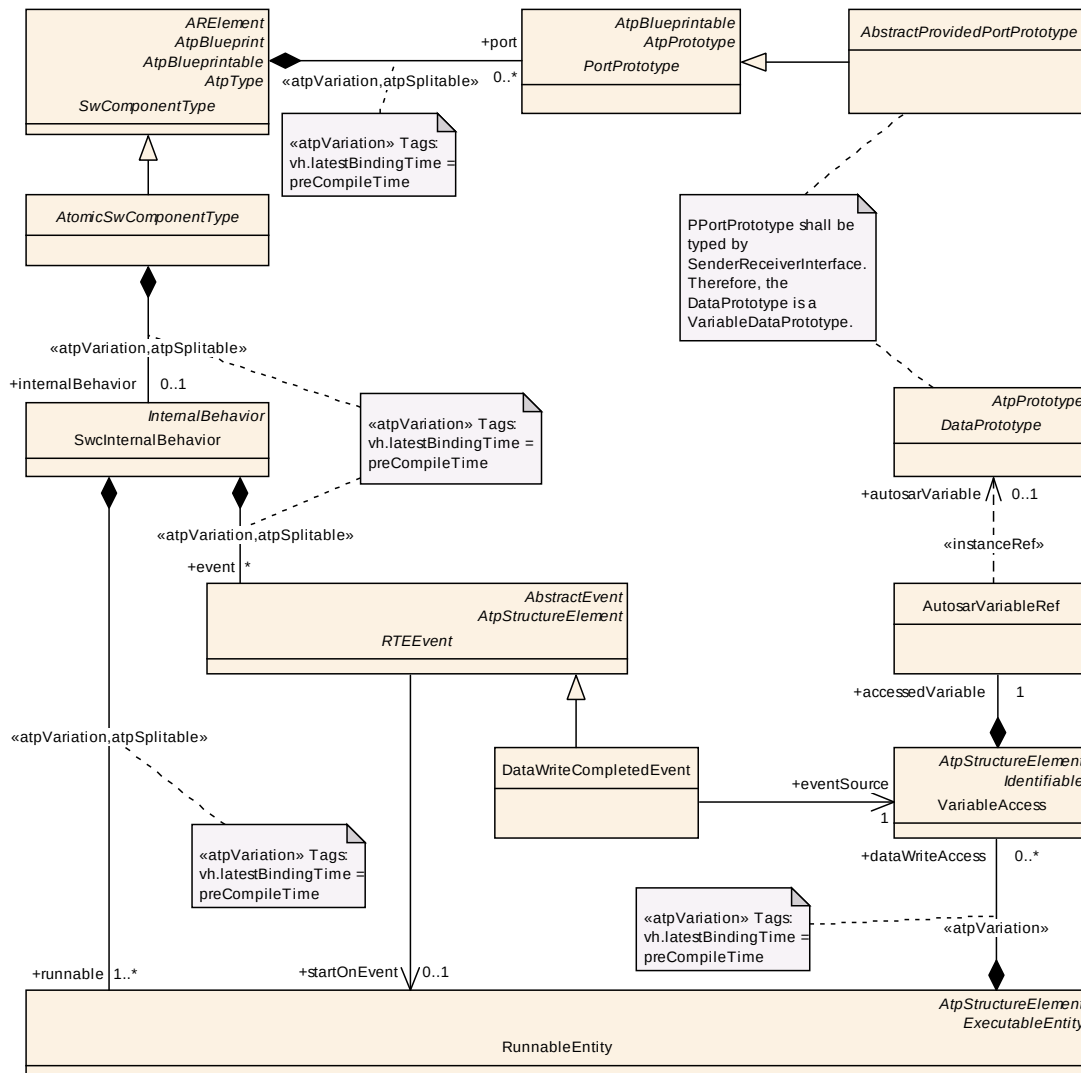


Figure 7.22: dataWriteAccess

7.5.1.6 DataReceivedEvent

[TPS_SWCT_01337] **DataReceivedEvent** [A receiver is notified through the same event mechanism when a **VariableDataPrototype** is received. As shown in Figure 7.23, the **DataReceivedEvent** is directly associated with the corresponding **VariableDataPrototype**.] (RS_SWCT_00200)

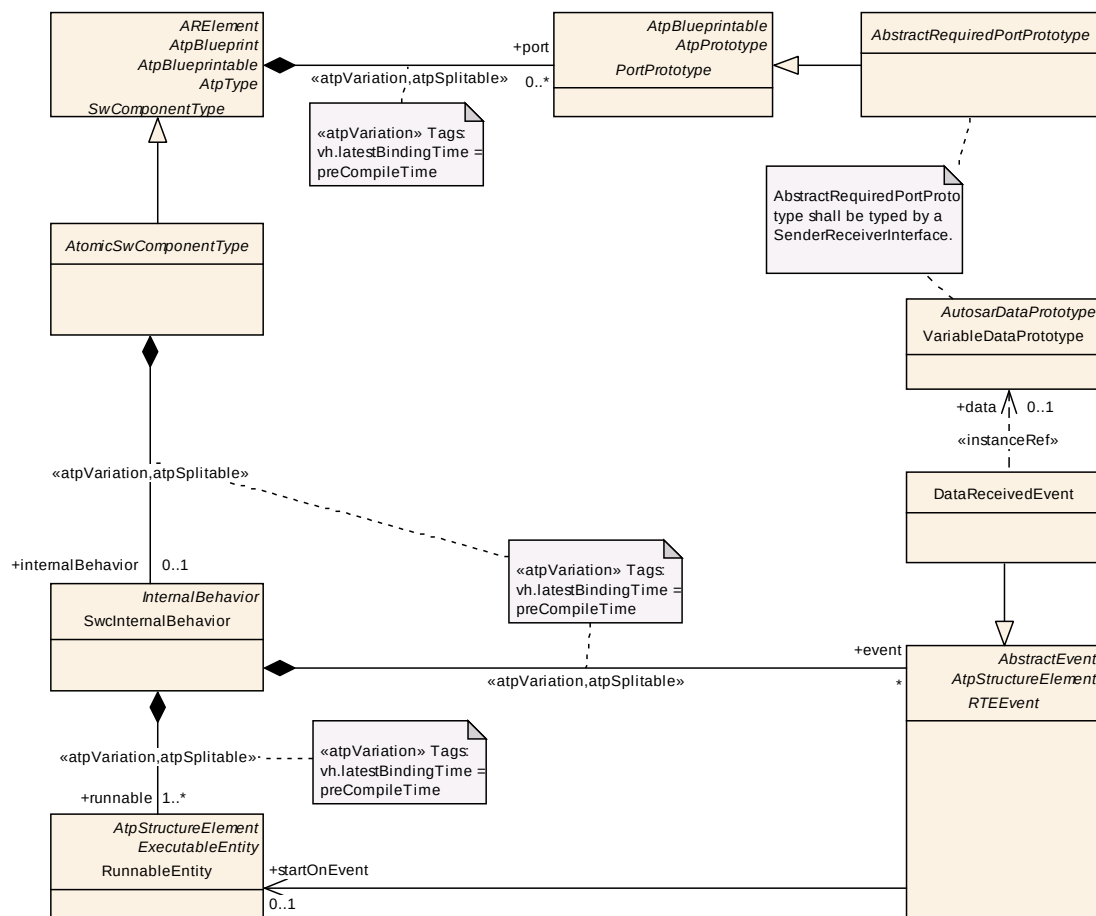


Figure 7.23: Receiver is notified by an event when new data has arrived

Class	DataReceivedEvent			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::RTE Events			
Note	The event is raised when the referenced data elements are received.			
Base	ARObject, AbstractEvent ,AtpClassifier,AtpFeature,AtpStructure Element, Identifiable ,MultilanguageReferrable, RTEEvent , Referrable			
Attribute	Datatype	Mul.	Kind	Note
data	VariableDataPrototype	0..1	iref	Data element referenced by event

Table 7.32: DataReceivedEvent

7.5.1.7 DataReceiveErrorEvent

[TPS_SWCT_01338] DataReceiveErrorEvent [A receiver is notified of [DataReceiveErrorEvent](#) through the activation of its [RunnableEntity](#) which is referenced by this [RTEEvent](#). A [DataReceiveErrorEvent](#) includes a reference to a [VariableDataPrototype](#) and is raised by the RTE when an error concerning

the reception of the referenced data is detected by the COM ⁵ layer. The following cases present some situations which will cause the RTE to raise a `DataReceiveErrorEvent`:

- the RTE receives a signal-outdated notification from the COM layer when a monitored periodic signal is not received in time. The COM layer monitors the validity of the signal's value based on the value of the `aliveTimeout` attribute of `ReceiverComSpec` referencing the `VariableDataPrototype` associated with the signal. If the time elapsed since the last update of a signal's value exceeds its `aliveTimeout` then the COM layer notifies the RTE of a signal outdated error.
- The RTE receives a signal invalid notification from the COM layer when the COM layer detects that an incoming signal has the predefined "invalid" value.

]([RS_SWCT_00200](#))

[TPS_SWCT_01339] RTE activates `RunnableEntity` in response to `DataReceiveErrorEvent` [A `DataReceiveErrorEvent` is used by the RTE to activate a `RunnableEntity` that is supposed to handle the above-mentioned errors.

The error code will be made available to the activated `RunnableEntity` through the appropriate RTE API function.]([RS_SWCT_00200](#))

[TPS_SWCT_01340] `DataReceiveErrorEvent` cannot be combined with a `WaitPoint` [Please note that a `DataReceiveErrorEvent` cannot be associated with a `WaitPoint`, see [[constr_1091](#)].

It can only be used for the receiver software-component in a sender-receiver communication and its data reference is restricted to `VariableDataPrototypes` with their `swImplPolicy` attribute not set to `queued`.]([RS_SWCT_00200](#))

⁵In case of internal communication the RTE is not enforced to use the COM layer. It is also possible to implement the required behavior directly in the RTE.

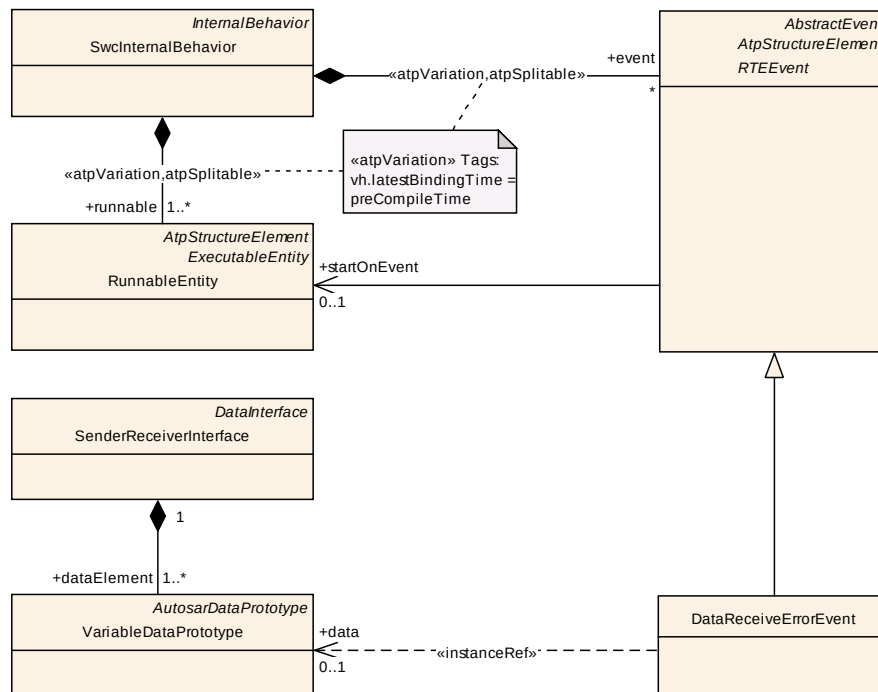


Figure 7.24: DataReceiveErrorEvent references a Runnable and a VariableDataPrototype

[TPS_SWCT_01341] **DataReceiveErrorEvent** is directly associated with the corresponding **VariableDataPrototype** [As shown in Figure 7.24, the **DataReceiveErrorEvent** is directly associated with the corresponding **VariableDataPrototype** and references the **RunnableEntity** that is activated due to the occurrence of this **RTEEvent**.] (RS_SWCT_00200)

Class	DataReceiveErrorEvent			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::RTE Events			
Note	This event is raised by the RTE when the Com layer detects and notifies an error concerning the reception of the referenced data element.			
Base	ARObject, AbstractEvent, AtpClassifier, AtpFeature, AtpStructure Element, Identifiable, MultilanguageReferrable, RTEEvent, Referrable			
Attribute	Datatype	Mul.	Kind	Note
data	VariableDataPrototype	0..1	iref	Data element referenced by event

Table 7.33: DataReceiveErrorEvent

7.5.2 RunnableEntities and Client Server Communication

7.5.2.1 Invoking an Operation

[TPS_SWCT_01342] **Invocation of a server operation** [A **RunnableEntity** invokes a server operation formally defined as a **ClientServerOperation** via an

`RPortPrototype` of the enclosing `SwComponentPrototype` typed by a particular `AtomicSwComponentType`. [\]\(RS_SWCT_00200\)](#)

[TPS_SWCT_01343] Synchronous vs. asynchronous invocation [\[](#) A `ClientServerOperation` itself can be invoked either “synchronously” or “asynchronously”. [\]\(RS_SWCT_00200\)](#)

In the majority of cases the `ClientServerOperation` will be invoked at a different `SwComponentPrototype` but in general it would be possible to invoke a `ClientServerOperation` on the same `SwComponentPrototype` as well.

The decision whether a specific `ClientServerOperation` is called synchronously or asynchronously needs to be specified in the formal description of the corresponding `AtomicSwComponentType`, namely in the context of an `SwcInternalBehavior` (see Figure 7.25 for more details).

But it is not supported to invoke the same instance of a `ClientServerOperation` synchronously and asynchronously together.

[constr_2022] Mutually exclusive use of `SynchronousServerCallPoints` and `AsynchronousServerCallPoints` [\[](#) A `ClientServerOperation` of a particular `RPortPrototype` shall be mutually exclusive referenced by either a `SynchronousServerCallPoints` or an `AsynchronousServerCallPoints`. [\]\(/\)](#)

[TPS_SWCT_01344] Consistency of values of `timeout` [\[](#) The `timeout` values need to be consistent in case of multiple `ServerCallPoints` referencing the same instance of `ClientServerOperation`. [\]\(RS_SWCT_00200\)](#)

[constr_2023] Consistency of `timeout` values [\[](#) The `timeout` values of all `ServerCallPoints` referencing the same instance of `ClientServerOperation` in a `RPortPrototype` shall be identical. [\]\(/\)](#)

[TPS_SWCT_01345] Synchronous operation invocation [\[](#) In case of a synchronous operation invocation the particular `RunnableEntity` merely needs a `SynchronousServerCallPoint` (see Figure 7.25). [\]\(RS_SWCT_00200\)](#)

[TPS_SWCT_01346] Asynchronous operation invocation [\[](#) Asynchronous invocation is a bit more complex because it is necessary to specify how to respond to a notification about the completion of the corresponding operation.

This is done using the generic `RTEEvent` mechanism: the notification about an asynchronously executed operation having completed is implemented as an `AsynchronousServerCallReturnsEvent`.

Therefore, if an `AsynchronousServerCallReturnsEvent` is raised the RTE can either trigger the execution of a specific `RunnableEntity` or the `AtomicSwComponentType` can implement a `WaitPoint` that blocks the execution of the calling `RunnableEntity` until the `AsynchronousServerCallReturnsEvent` is recognized. [\]\(RS_SWCT_00200\)](#)

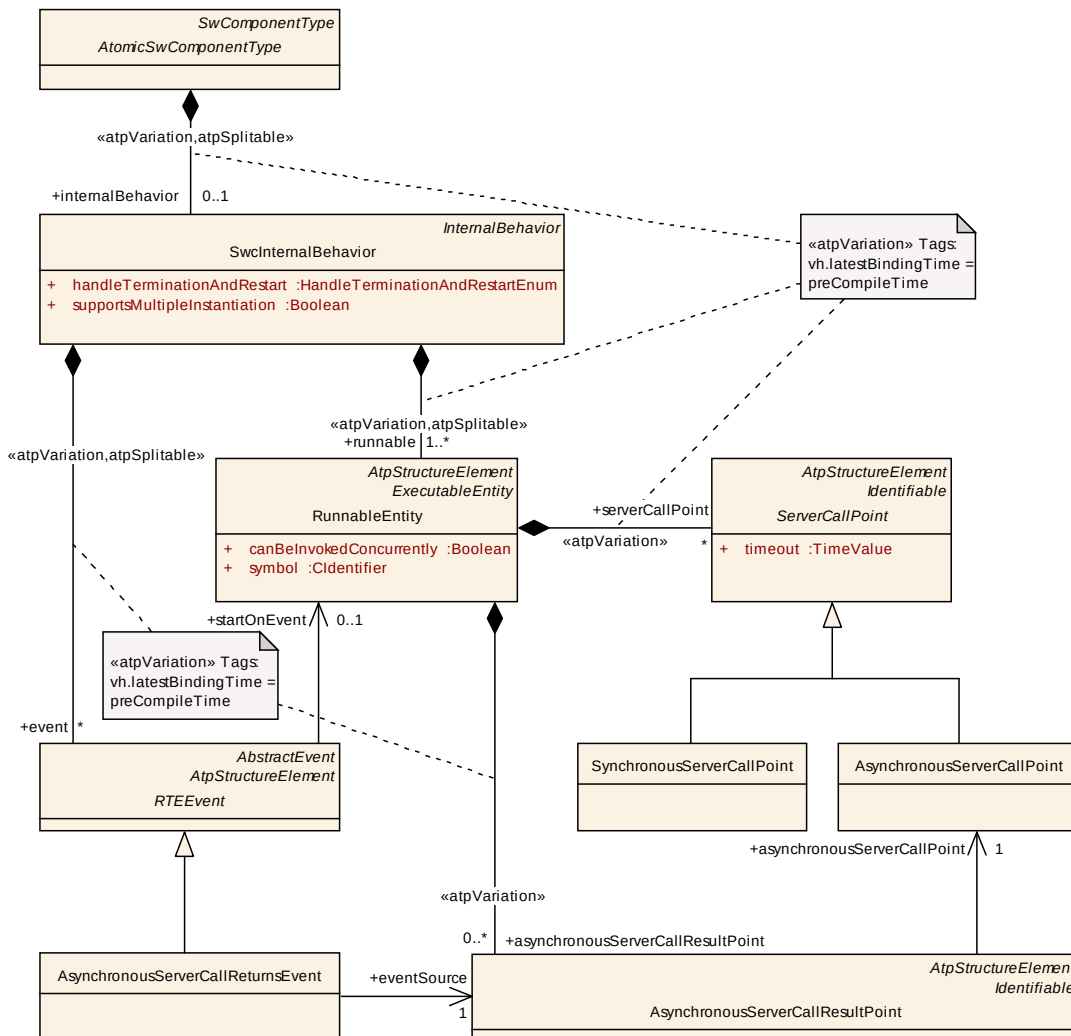


Figure 7.25: Model of a server call point.

For example, let's consider the case of an asynchronous call to a remote operation where the RTE is supposed to trigger a specific **RunnableEntity** when the operation completes. The description of the corresponding **AtomicSwComponentType** would typically contain the following elements:

1. The **AtomicSwComponentType** contains an **RPortPrototype** 'myPort' typed by a **PortInterface** that in turn contains the definition of an **ClientServerOperation** 'remoteOperation'.
2. The **AtomicSwComponentType**'s **SwcInternalBehavior** contains at least two **RunnableEntity**s: the **RunnableEntity** 'main' is supposed to invoke the operation; the **RunnableEntity** 'callback' is the one that should be called when the operation completes.
3. The description of the **RunnableEntity** 'main' contains an **AsynchronousServerCallPoint** 'invokeMyOperation' referencing the respective **ClientServerOperation** in the **PortInterface** used to type the **PortPro**

otype 'myPort'. This implies that the `RunnableEntity` is allowed to invoke this operation asynchronously.

4. The description of the `RunnableEntity` 'callback' contains an `AsynchronousServerCallResultPoint` 'fetchMyOperationResults' referencing the respective `AsynchronousServerCallPoint` 'invokeMyOperation'. This implies that the `RunnableEntity` is allowed to fetch the results of the asynchronously invoked operation.
5. The description of the `SwcInternalBehavior` includes an `AsynchronousServerCallReturnsEvent` 'myOperationReturns' which references the previously defined `AsynchronousServerCallResultPoint` 'fetchMyOperationResults'.
6. The description of the `AsynchronousServerCallReturnsEvent` 'myOperationReturns' references the `RunnableEntity` 'callback', indicating that the RTE should trigger the execution of this Runnable when 'myOperationReturns' is raised.

Class	ServerCallPoint (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::ServerCall			
Note	If a <code>RunnableEntity</code> owns a <code>ServerCallPoint</code> it is entitled to invoke a particular <code>ClientServerOperation</code> of a specific <code>RPortPrototype</code> of the corresponding <code>AtomicSwComponentType</code>			
Base	<code>ARObject</code> , <code>AtpClassifier</code> , <code>AtpFeature</code> , <code>AtpStructureElement</code> , <code>Identifiable</code> , <code>Multilanguage</code> , <code>Referrable</code> , <code>Referrable</code>			
Attribute	Datatype	Mul.	Kind	Note
operation	<code>ClientServerOperation</code>	0..1	iref	The operation that is called by this runnable.
timeout	<code>TimeValue</code>	1	attr	Time in seconds before the server call times out and returns with an error message. It depends on the call type (synchronous or asynchronous) how this is reported.

Table 7.34: ServerCallPoint

Class	SynchronousServerCallPoint			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::ServerCall			
Note	This means that the <code>RunnableEntity</code> is supposed to perform a blocking wait for a response from the server.			
Base	<code>ARObject</code> , <code>AtpClassifier</code> , <code>AtpFeature</code> , <code>AtpStructureElement</code> , <code>Identifiable</code> , <code>Multilanguage</code> , <code>Referrable</code> , <code>Referrable</code> , <code>ServerCallPoint</code>			
Attribute	Datatype	Mul.	Kind	Note
calledFromWithinExclusiveArea	<code>ExclusiveAreaNestingOrder</code>	0..1	ref	This indicates that the call point is located at the deepest level inside one or more <code>ExclusiveAreas</code> that are nested in the given order.

Table 7.35: SynchronousServerCallPoint

Class	AsynchronousServerCallPoint			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::ServerCall			
Note	An AsynchronousServerCallPoint is used for asynchronous invocation of a ClientServerOperation. IMPORTANT: a ServerCallPoint cannot be used concurrently. Once the client RunnableEntity has made the invocation, the ServerCallPoint cannot be used until the call returns (or an error occurs!) at which point the ServerCallPoint becomes available again.			
Base	ARObject,AtpClassifier,AtpFeature,AtpStructureElement,Identifiable,Multilanguage Referrable,Referrable,ServerCallPoint			
Attribute	Datatype	Mul.	Kind	Note
–	–	–	–	–

Table 7.36: AsynchronousServerCallPoint

Class	AsynchronousServerCallResultPoint			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::ServerCall			
Note	If a RunnableEntity owns a AsynchronousServerCallResultPoint it is entitled to get the result of the referenced AsynchronousServerCallPoint. If it is associated with AsynchronousServerCallReturnsEvent, this RTEEvent notifies the completion of the required ClientServerOperation or a timeout. The occurrence of this event can either unblock a WaitPoint or can lead to the invocation of a RunnableEntity.			
Base	ARObject,AtpClassifier,AtpFeature,AtpStructureElement,Identifiable,Multilanguage Referrable,Referrable			
Attribute	Datatype	Mul.	Kind	Note
asynchron ousServer CallPoint	AsynchronousS erverCallPoint	1	ref	The referenced Asynchronous Server Call Point defines the asynchronous server call from which the results are returned.

Table 7.37: AsynchronousServerCallResultPoint

[constr_2006] Number of [AsynchronousServerCallResultPoint](#) referencing to one [AsynchronousServerCallPoint](#) [The [AsynchronousServerCallPoint](#) has to be referenced by exactly one [AsynchronousServerCallResultPoint](#). This means that only the [RunnableEntity](#) with this [AsynchronousServerCallResultPoint](#) can fetch the result of the asynchronous server invocation of this particular [AsynchronousServerCallPoint](#).]()

This information might be used by the RTE generator to optimize the data consistency mechanisms.

Class	AsynchronousServerCallReturnsEvent			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::RTE Events			
Note	This event is raised when an asynchronous server call is finished.			
Base	ARObject,AbstractEvent,AtpClassifier,AtpFeature,AtpStructure Element,Identifiable,MultilanguageReferrable,RTEEvent,Referrable			
Attribute	Datatype	Mul.	Kind	Note

Attribute	Datatype	Mul.	Kind	Note
eventSource	AsynchronousServerCallResultPoint	1	ref	The referenced AsynchronousServerCallResultPoint which is raises the RTEEvent in case of returning asynchronous server call.

Table 7.38: AsynchronousServerCallReturnsEvent

[TPS_SWCT_01347] Blocking access to operation result in an asynchronous operation invocation [If the call of the RTE fetching the operations results shall block until the server returns the RunnableEntity with the AsynchronousServerCallResultPoint needs additional a WaitPoint referencing the AsynchronousServerCallReturnsEvent which is associated with the AsynchronousServerCallResultPoint representing the operations results access.

In this case the AsynchronousServerCallReturnsEvent shall not define a startOnEvent reference to a RunnableEntity.](RS_SWCT_00200)

[constr_2030] AsynchronousServerCallResultPoint combined with WaitPoint shall belong to the same RunnableEntity [The WaitPoint which references a AsynchronousServerCallReturnsEvent and the AsynchronousServerCallResultPoint which is referenced by this AsynchronousServerCallReturnsEvent shall be aggregated by the same RunnableEntity.]()

7.5.2.2 Providing an Implementation of an Operation

A software-component can define an OperationInvokedEvent for each operation inside one of the server AbstractProvidedPortPrototypes. This way a RunnableEntity may respond to such an invocation through the generic event handling mechanisms described above (as formally expressed in Figure 7.26).

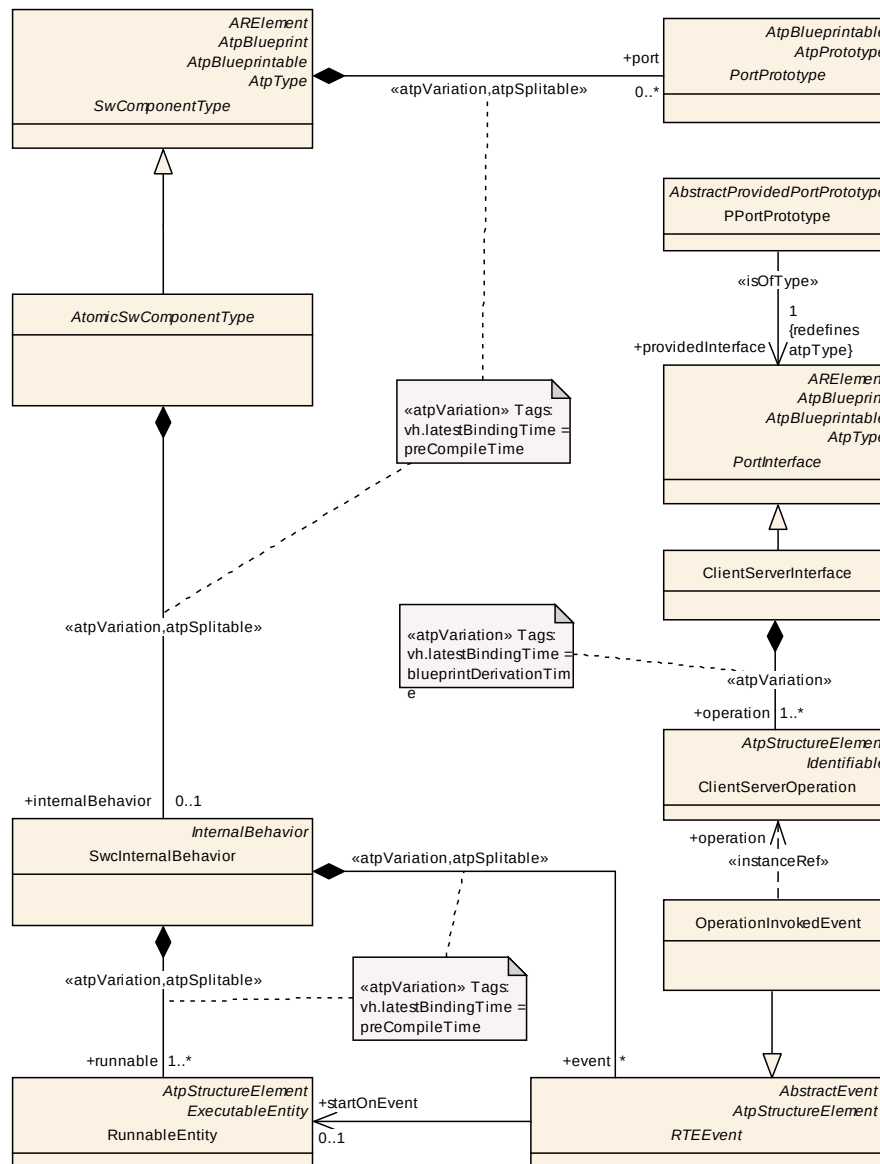


Figure 7.26: The `OperationInvokedEvent` references the operation that was called by a client.

Class	OperationInvokedEvent			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::RTE Events			
Note	The OperationInvokedEvent references the ClientServerOperation invoked by the client.			
Base	ARObject, AbstractEvent , AtpClassifier, AtpFeature, AtpStructure Element, Identifiable , MultilanguageReferrable, RTEEvent , Referrable			
Attribute	Datatype	Mul.	Kind	Note
operation	ClientServerOperation	0..1	iref	The operation to be executed as the consequence of the event.

Table 7.39: OperationInvokedEvent

7.5.2.3 Reacting on Data Transformation Errors

[TPS_SWCT_01624] Hard error occurs during the execution of a transformer chain [If a hard error occurs during the execution of a transformer chain which is executed

- on the server side of a client/server communication and re-transforms the data which trigger a server [RunnableEntity](#) or
- on the trigger sink side of an inter-ECU external trigger communication,

this server [RunnableEntity](#) or trigger sink [RunnableEntity](#) cannot be started because the re-transformed data are not available.]([RS_SWCT_03222](#))

This might be a problem for the software-component if the software-component wants to react on transformer errors.

[TPS_SWCT_01616] Semantics of [TransformerHardErrorEvent](#) [A software-component can define a [TransformerHardErrorEvent](#)

- for each [ClientServerOperation](#) inside one of the server [PPortPrototypes](#) (i.e. typed by a [ClientServerInterface](#)) or
- for each [Trigger](#) in trigger sink [RPortPrototypes](#) (i.e. typed by a [TriggerInterface](#)).

This way, a given [RunnableEntity](#) may define its response to a transformer error.]([RS_SWCT_03222](#))

Class	TransformerHardErrorEvent			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::RTE Events			
Note	The event is raised when data are received which should trigger a Client/Server operation or an external trigger but during transformation of the data a hard transformer error occurred.			
Base	ARObject, AbstractEvent ,AtpClassifier,AtpFeature,AtpStructureElement, Identifiable ,MultilanguageReferrable, RTEEvent , Referrable			
Attribute	Datatype	Mul.	Kind	Note
operation	ClientServerOperation	0..1	iref	This represents the ClientServerOperation to which the TransformerHardErrorEvent refers to.
trigger	Trigger	0..1	iref	Trigger for which the transformer can trigger this TransformerHardErrorEvent

Table 7.40: TransformerHardErrorEvent

7.5.3 RunnableEntities and External Trigger Event Communication

7.5.3.1 Trigger Source

[TPS_SWCT_01348] Trigger source [A [RunnableEntity](#) of the triggering software-component raises an external trigger event via an [AbstractProvidedPortPrototype](#) of the enclosing [SwComponentPrototype](#) typed by a particular [AtomicSwComponentType](#).

For this purpose the particular [RunnableEntity](#) needs an [ExternalTriggeringPoint](#) that references the particular instance of the trigger in a [PPortPrototype](#).
]([RS_SWCT_00200](#))

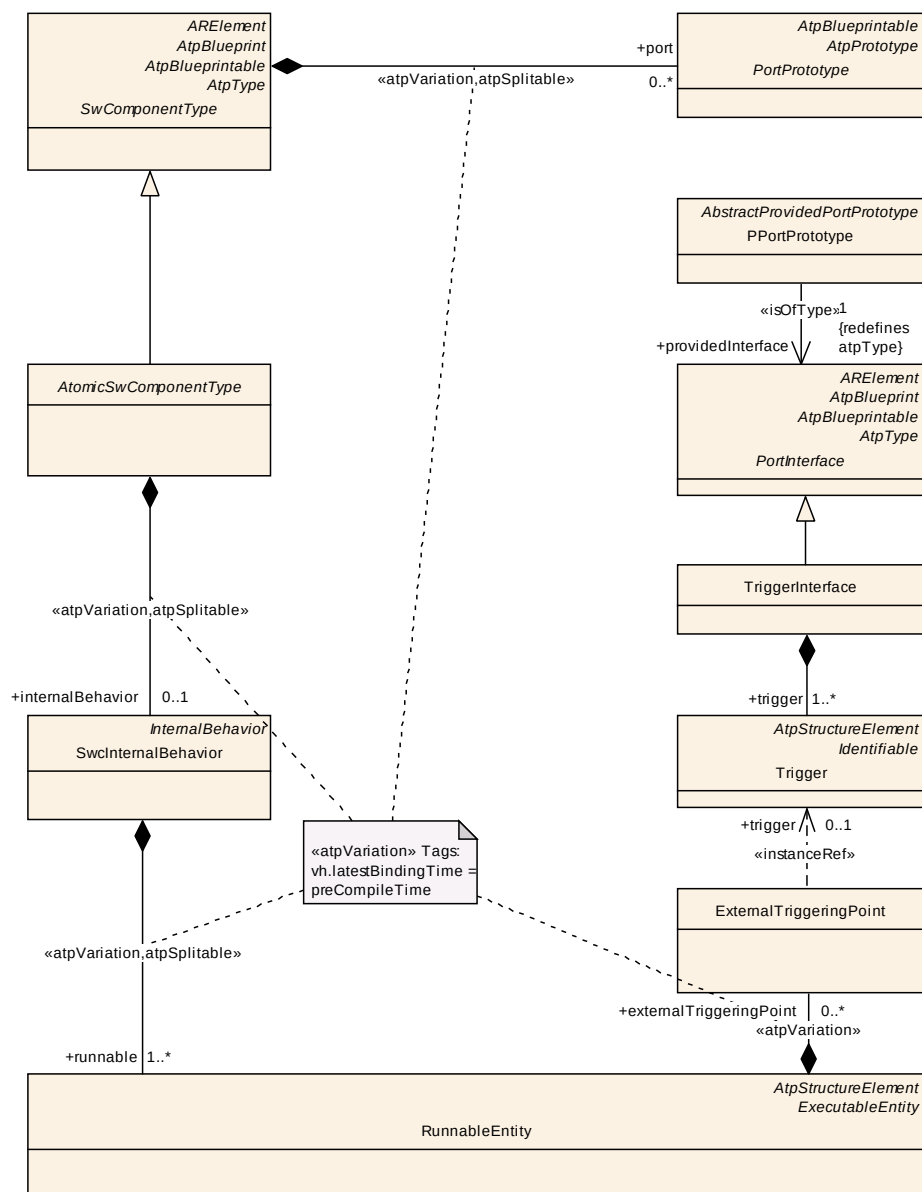


Figure 7.27: Model structure of a trigger source.

Class	ExternalTriggeringPoint			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::Trigger			
Note	If a RunnableEntity owns an ExternalTriggeringPoint it is entitled to raise an ExternalTriggerOccurredEvent.			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
ident	ExternalTriggeringPointIdent	0..1	aggr	The aggregation in the role ident provides the ability to make the ExternalTriggeringPoint identifiable. From the semantical point of view, the ExternalTriggeringPoint is considered a first-class Identifiable and therefore the aggregation in the role ident shall always exist (until it may be possible to let ModeAccessPoint directly inherit from Identifiable). Tags: atp.Status=shallBecomeMandatory xml.sequenceOffset=-100
trigger	Trigger	0..1	iref	The trigger taken for the ExternalTriggeringPoint. Tags: xml.namePlural=TRIGGER-IREF; xml.roleElement=false; xml.roleWrapperElement=true; xml.typeElement=true; xml.typeWrapperElement=false

Table 7.41: ExternalTriggeringPoint

7.5.3.2 Trigger Sink

The activation of [RunnableEntity](#) in the trigger sink is effected through the generic event handling mechanism.

[TPS_SWCT_01349] Trigger sink [The fact that a [RunnableEntity](#) shall be activated on occurrence of an external trigger event is formally defined by means of [ExternalTriggerOccurredEvent](#) that references a particular instance of the trigger in a [RPortPrototype](#) and additionally the [RunnableEntity](#) to be executed in response to the event.]([RS_SWCT_00200](#))

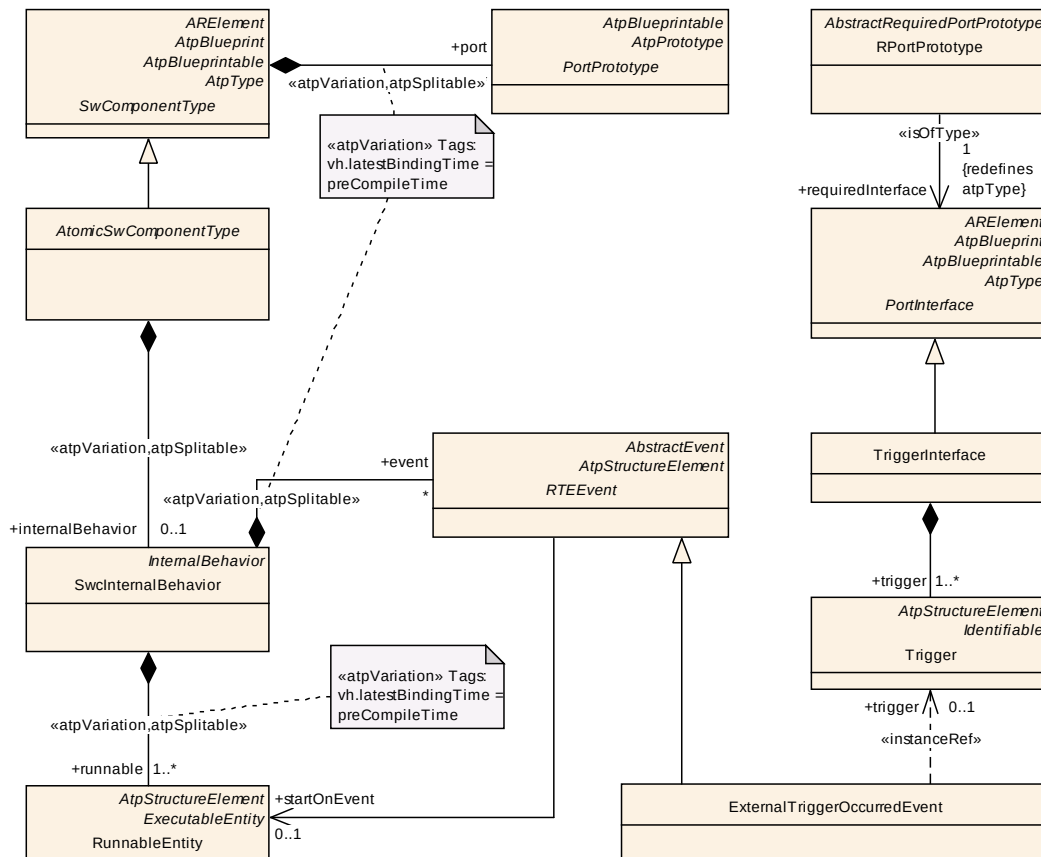


Figure 7.28: Model structure of a trigger sink

Class	ExternalTriggerOccurredEvent			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::RTE Events			
Note	The event is raised when the referenced trigger have been occurred.			
Base	ARObject, AbstractEvent , AtpClassifier, AtpFeature, AtpStructure Element, Identifiable , MultilanguageReferrable, RTEEvent , Referrable			
Attribute	Datatype	Mul.	Kind	Note
trigger	Trigger	0..1	iref	Reference to the applicable Trigger.

Table 7.42: ExternalTriggerOccurredEvent

7.5.4 RunnableEntities and Parameter Access

There are several ways a Calibration Parameter is provided within a software component.

[TPS_SWCT_01350] Calibration Parameters shared among several `SwComponentTypes` | As mentioned above, if Calibration Parameters are shared among several `SwComponentTypes` a dedicated `PortInterface` in a `PortPrototype` will be used. | (*RS SWCT 00200*)

The designer of a software-component can use this access mechanism when designing a `RunnableEntity` using, as input value, a `DataPrototype`

- from an arbitrary `RPortPrototype` associated with a `ClientServerInterface`, `SenderReceiverInterface` or a `NvDataInterface`,
- `VariableDataPrototype` in the context of an `SwcInternalBehavior`

This input value will be fed to an interpolation routine whose result can be used internally or transferred to a adjacent `SwComponentPrototype` via dedicated `PortPrototypes`. Typically, there will be a dedicated `RunnableEntity` (with “ReceiveMode” set to “activation_of_runnable_entity”) that itself calls the interpolation routine with the appropriate input value and the appropriate `ParameterDataPrototype`.

Note that the `ParameterAccess` also allows to set input values or shared axis through `SwDataDefProps` which are specific to the access point.

The result of this interpolation routine call is provided as an `ArgumentDataPrototype` with `direction` being either set to `out` or `inout` in a `ClientServerInterface`.

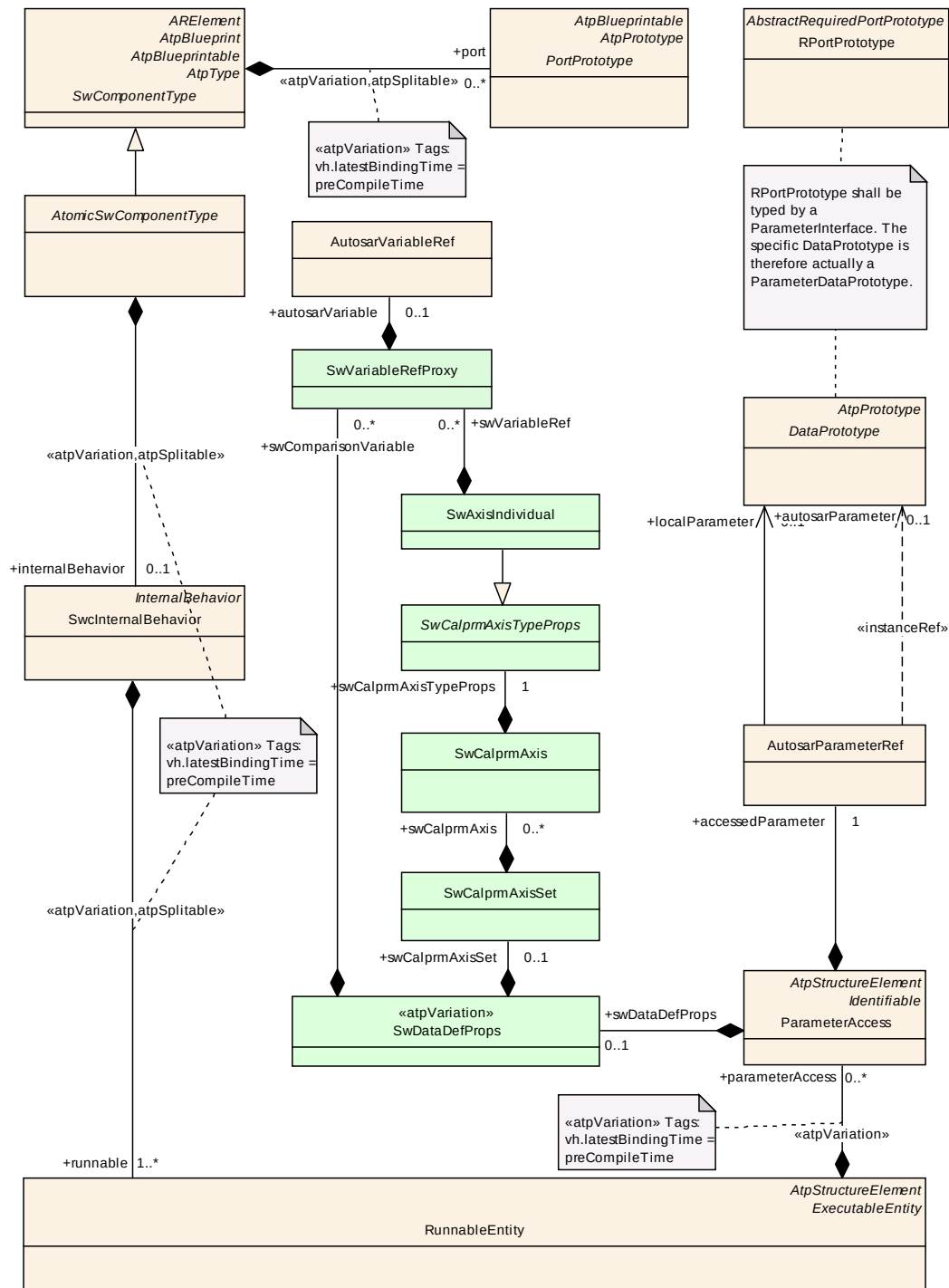


Figure 7.29: Runnable Access to a Calibration Port

Class	ParameterAccess			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::Data Elements			
Note	The presence of a ParameterAccess implies that a RunnableEntity needs access to a ParameterDataPrototype.			
Base	ARObject, AtpClassifier, AtpFeature, AtpStructureElement, Identifiable , Multilanguage Referrable, Referrable			
Attribute	Datatype	Mul.	Kind	Note
accessedParameter	AutosarParameterRef	1	aggr	Refernce to the accessed calibration parameter.
swDataDefProps	SwDataDefProps	0..1	aggr	This allows denote instance and access specific properties, mainly input values and common axis.

Table 7.43: ParameterAccess

[TPS_SWCT_01351] Access to a [ParameterDataPrototype](#) [The access to a [ParameterDataPrototype](#) will be indicated

- by the [ParameterAccess](#) entity if the [RunnableEntity](#) wants to access it from a [RPortPrototype](#). This is shown in Figure 7.29
- by defining the [ParameterAccess](#) association from a [RunnableEntity](#) to the [ParameterDataPrototype](#) in the roles [sharedParameter](#) or [perInstanceParameter](#). This is shown in Figure 2.3 in the lower association from [RunnableEntity](#) to [ParameterDataPrototype](#)

]([RS_SWCT_00200](#))

Note: A [ParameterDataPrototype](#) in the roles [constantMemory](#) is not provided by the RTE and therefore the [ParameterAccess](#) association is not required to control the RTE API generation.

7.5.4.1 InstantiationDataDefProps

Typically, the accessibility and further information like alias names for a particular piece of data is modeled on the level of [DataPrototypes](#) (especially [VariableDataPrototypes](#), [ParameterDataPrototypes](#)).

But due to the recursive structure of the meta-model concerning data types (an [ApplicationCompositeDataType](#) consists of [DataPrototypes](#)), a part of the relevant MCD information is described directly in the data type (in case of a [ApplicationCompositeDataType](#)).

This is a strong restriction in the reuse of data types because the [ApplicationCompositeDataType](#) should be re-used for different [VariableDataPrototypes](#) and [ParameterDataPrototypes](#) to guarantee type compatibility on C-implementation level (e.g. data of a [PortPrototype](#) is stored in a PIM or a [ParameterDataPrototype](#) used as ROM Block and shall be typed by the same data type as NVRAM Block).

This restriction is overcome by [InstantiationDataDefProps](#) as shown in figure 7.30

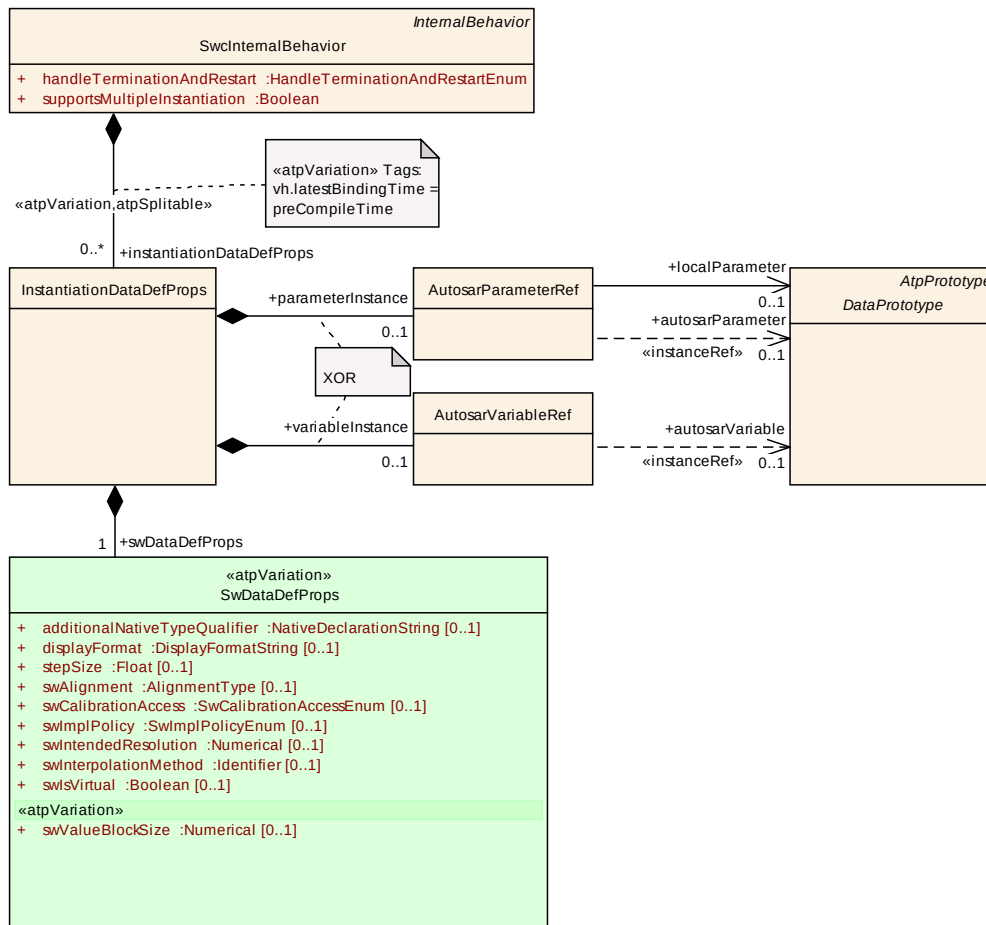


Figure 7.30: applying instantiation specific data definition properties

Class	InstantiationDataDefProps			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::InstantiationDataDefProps			
Note	This is a general class allowing to apply additional SwDataDefProps to particular instantiations of a DataPrototype. Typically the accessibility and further information like alias names for a particular data is modeled on the level of DataPrototypes (especially VariableDataPrototypes, ParameterDataPrototypes). But due to the recursive structure of the meta-model concerning data types (a composite (data) type consists out of data prototypes) a part of the MCD information is described in the data type (in case of ApplicationCompositeDataType). This is a strong restriction in the reuse of data typed because the data type should be re-used for different VariableDataPrototypes and ParameterDataPrototypes to guarantee type compatibility on C-implementation level (e.g. data of a Port is stored in PIM or a ParameterDataPrototype used as ROM Block and shall be typed by the same data type as NVRAM Block). This class overcomes such a restriction if applied properly.			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
parameterInstance	AutosarParameterRef	0..1	aggr	This is the particular ParameterDataPrototypes on which the swDataDefProps shall be applied.
swDataDefProps	SwDataDefProps	1	aggr	These are the particular data definition properties which shall be applied
variableInstance	AutosarVariableRef	0..1	aggr	This is the particular VariableDataPrototypes on which the swDataDefProps shall be applied.

Table 7.44: InstantiationDataDefProps

7.5.5 RunnableEntities and Mode Communication

For the communication of modes between [RunnableEntities](#) we have to distinguish between two use cases.

[TPS_SWCT_01352] Requested mode is just sent and received as an ordinary data value [In the first case, a requested mode is just sent and received as an ordinary data value without specifying the details of mode switching in the corresponding port interface.

This mechanism is used if the receiving [RunnableEntity](#) is not directly implementing a mode switch but does further processing of the mode request. This is especially needed to transfer mode requests between ECUs.

In this case, the mode is transferred via sender-receiver communication so that the involved [RunnableEntities](#) just need the same type of APIs against the RTE as for sender-receiver communication.

This is possible, because `ModeDeclarationGroupPrototypes` can be mapped to an `ImplementationDataTypes`. This concept and the meta-classes needed for the mapping are further explained in chapter 4.2.5. [|\(RS_SWCT_00200\)](#)

[TPS_SWCT_01353] RunnableEntitys react on a mode request via a corresponding RTEEvent [|](#) In the second case, one `RunnableEntity` “sends” a mode request and one or more other `RunnableEntitys` react on the request via a corresponding `RTEEvent` or by being suppressed from being triggered any longer by other `RTEEventS`.

In this case, special APIs against the RTE are required and the RTE has to implement the actual mode switch. This kind of communication is only possible between software-components on the same ECU. For further explanation of the general concept refer to chapter 4.2.5 and for the details of the meta-model for mode switches refer to chapter 9. [|\(RS_SWCT_00200, RS_SWCT_03202\)](#)

7.6 Port API Options

[TPS_SWCT_01354] PortAPIOption [|](#) The RTE Generator needs additional options per `PortPrototype` to choose the proper generation schema. These are subsumed in the `PortAPIOption` element which is shown in Figure 7.31. [|\(\)](#)

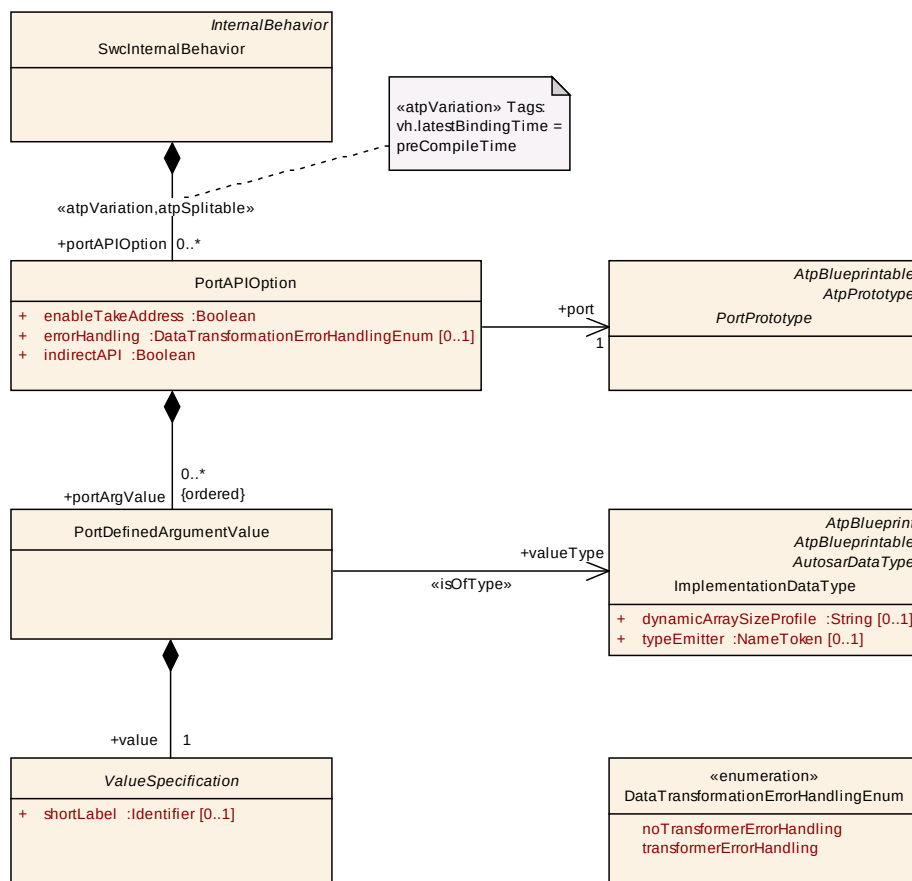


Figure 7.31: Port API Options.

Class	PortAPIOption			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::PortAPIOptions			
Note	Options how to generate the signatures of calls for an AtomicSwComponentType in order to communicate over a PortPrototype (for calls into a RunnableEntity as well as for calls from a RunnableEntity to the PortPrototype).			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
enableTakeAddress	Boolean	1	attr	If set to true, the software-component is able to use the API reference for deriving a pointer to an object.
errorHandling	DataTransformationErrorHandlingEnum	0..1	attr	This specifies whether the RunnableEntitys which access a PortPrototype that it referenced by this PortAPIOption shall specifically handle transformer errors or not.
indirectAPI	Boolean	1	attr	If set to true this attribute specifies an "indirect API" to be generated for the associated port which means that the SWC is able to access the actions on a port via a pointer to an object representing a port. This allows e.g. iterating over ports in a loop. This option has no effect for PPortPrototypes of client/server interfaces.
port	PortPrototype	1	ref	The option is valid for generated functions related to communication over this port
portArgumentValue (ordered)	PortDefinedArgumentValue	*	aggr	An argument value defined by this port.

Table 7.45: PortAPIOption

[TPS_SWCT_01626] Error notification of data transformer errors [If the attribute [PortAPIOption.errorHandling](#) is set to [transformerErrorHandling](#) then all [RunnableEntitys](#) accessing the [PortPrototype](#) referenced by [port](#) shall handle the extended transformer error notification.] ([RS_SWCT_03222](#))

Enumeration	DataTransformationErrorHandlingEnum
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::PortAPIOptions
Note	This enumeration defines different ways how runnables shall handle transformer errors.
Literal	Description
noTransformerErrorHandling	A runnable does not handle transformer errors.
transformerErrorHandling	The runnable implements the handling of transformer errors.

Table 7.46: DataTransformationErrorHandlingEnum

7.6.1 Enable to Take Address

[TPS_SWCT_01355] `enableTakeAddress` = true [If the attribute `enableTakeAddress` = true the generated API related to this `PortPrototype` is provided in a way that the software-component is able to use the API reference for deriving a pointer to an object.]()

The main focus of the feature is support for configuration of AUTOSAR Services which are limited to single instances.

[constr_2024] `enableTakeAddress` is restricted to single instantiation [The definition of a `PortAPIOption` with `enableTakeAddress` set to true is only permitted for software-components where the attribute `SwcInternalBehavior.supportsMultipleInstantiation` is set to false.]()

7.6.2 Indirect API Generation

[TPS_SWCT_01356] `indirectAPI` option switches the generation of the RTE's indirect API functionality [The `indirectAPI` option switches the generation of the RTE's indirect API functionality for a certain `PortPrototype`. The generated indirect API does allow to iterate over ports within the SW-Component.]()

7.6.3 Port Defined Argument Value

[TPS_SWCT_01357] Definition of implicit values that are passed by the RTE to the server's entry point [In addition to the formal parameters of a client/server invocation that are defined as part of the server's `PortInterface`, it is possible to specify a number of implicit values that are passed by the RTE to the server's entry point.]()

The initial need for this feature arises in the context of basic software services - although it is not limited to those.

For a service like the NVRAM manager, every accessing port is in addition to its logical identity - as a sequence of `shortNames` - uniquely identified through a NVRAM specific memory block id. This block id shall be defined in the context of ECU integration and not by the client components.

Instead of exposing this mechanism on the logical `ClientServerInterface` level in form of a formal `argument`, one or more `PortDefinedArgumentValues` can be specified.

[TPS_SWCT_01358] Values are hidden from the client components [Because these values are specified in the context of the provide-port only they are hidden from the client components keeping their design and code independent from the server component details.]()

In the example of the NVRAM manager, this allows to define the block id in the context of ECU integration and not by the client components.

Figure 7.31 shows the meta-model of Port API Options and the `portArgValue`.

[constr_1150] Usage of `valueType` for `PortDefinedArgumentValue` [The `valueType` (typically this boils down to integer values used to specify an “id”) associated with `PortDefinedArgumentValue` shall be of `category` `VALUE` or `TYPE_REFERENCE`. The latter case is only supported if the value of `category` of the target data type is set to `VALUE`.]()

In case of a `PPortPrototype` of the NVRAM example this list would have just one value of type `int8` or `int16` holding the memory block id.

[constr_1386] `PortDefinedArgumentValue` shall only be defined for `AbstractProvidedPortPrototype` [A `PortAPIOption` which aggregates at least one `PortDefinedArgumentValue` in the role `portArgValue` shall reference an `AbstractProvidedPortPrototype` typed by a `ClientServerInterface` in the role `port`.]()

To be clear, this means that `PortDefinedArgumentValues` may not be used together with `RPortPrototypes`.

Class	PortDefinedArgumentValue			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::PortAPIOptions			
Note	A <code>PortDefinedArgumentValue</code> is passed to a <code>RunnableEntity</code> dealing with the <code>ClientServerOperations</code> provided by a given <code>PortPrototype</code> . Note that this is restricted to <code>PPortPrototypes</code> of a <code>ClientServerInterface</code> .			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
value	ValueSpecification	1	aggr	Specifies the actual value.
valueType	ImplementationDataType	1	tref	The implementation type of this argument value. It should not be composite type or a pointer. Stereotypes: <code>isOfType</code>

Table 7.47: PortDefinedArgumentValue

7.7 PerInstanceMemory

[TPS_SWCT_01359] Private memory per instance [[AtomicSwComponentTypes](#) that support multiple instantiation (attribute `supportsMultipleInstantiation == true`) will typically need a given amount of private memory per instance. It is the responsibility of the RTE to provide a mechanisms with which each instance of an [AtomicSwComponentType](#) can access its own instance-specific memory.]()

[TPS_SWCT_01360] Arbitrary number of per-instance memory blocks [An [AtomicSwComponentType](#) can define an arbitrary number of per-instance memory blocks.]()

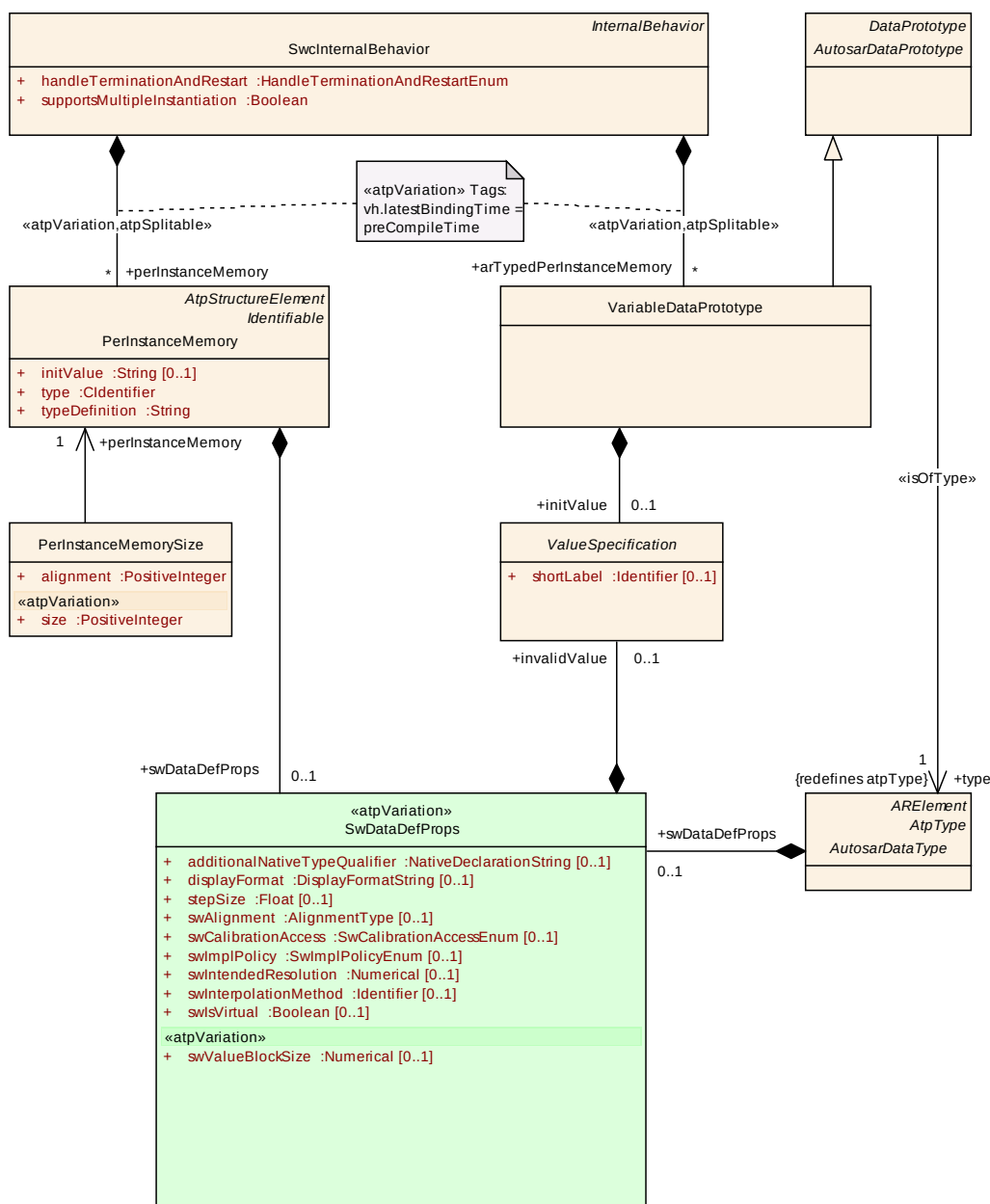


Figure 7.32: PerInstanceMemory

[TPS_SWCT_01361] attribute `supportsMultipleInstantiation == false` [`AtomicSwComponentTypes` that do *not* support multiple instantiation (attribute `supportsMultipleInstantiation == false`) do not necessarily need to use the `PerInstanceMemory`: because there will only be a single instance of the `AtomicSwComponentType` on an ECU, the `AtomicSwComponentType` can use static variables to store the `AtomicSwComponentType`'s internal state.

However, the usage of `PerInstanceMemory` is also allowed in this case. `]()`

[TPS_SWCT_01362] Initialization of `PerInstanceMemory` [Note that the `PerInstanceMemory` is not initialized by the RTE if no `initValue` is defined. In this case, it is the responsibility of the `AtomicSwComponentType` to initialize the `PerInstanceMemory`. `]()`

7.7.1 `PerInstanceMemory` typed by “C” Data Types

[TPS_SWCT_01363] `PerInstanceMemory` typed by “C” Data Types [For each such memory block, the software-component description shall provide the name of the data type (the “C”-type) it needs to store in the memory block in the attribute `type`.

This attribute allows for the RTE to generate an API function that provides a convenient and type-safe access to the data item.

In addition, the software-component description shall define the data type in the attribute `typeDefinition`. This attribute is supposed to contain a C typedef of the data type in valid C-syntax. `]()`

In other words, this `typeDefinition` shall be formulated such that it can be included verbatim in a C header file.

[constr_2007] Consistency of `typeDefinition` attribute [All `PerInstanceMemory`s of the same `SwcInternalBehavior` with identical `type` attribute shall define an identical `typeDefinition` attribute as well. `]()`

[TPS_SWCT_01364] Initial value of a `PerInstanceMemory` typed by “C” Data Types [The `initValue` is a comma separated list which can be used verbatim by the RTE generator as constant initializer. `]()`

[TPS_SWCT_01574] `PerInstanceMemory.typeDefinition` shall not contain a function pointer [The attribute `PerInstanceMemory.typeDefinition` is not allowed to contain a function pointer. `]()`

Please note that, although **[TPS_SWCT_01574]** is formulated like a constraint and the statement that it makes certainly has a constraint-ish nature, there is hardly a way to actually **enforce** the regulation because the content of `PerInstanceMemory.typeDefinition` is non-formal (modeled by the non-specific i.e. `String`).

Therefore, a specification item has been used for the description of the respective semantics rather than a constraint.

More details on the use of these attributes in the generation of software-component header-files can be found in the RTE specification [2].

Class	PerInstanceMemory			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::PerInstanceMemory			
Note	Defines a 'C' typed memory-block that needs to be available for each instance of the SW-component. This is typically only useful if supportsMultipleInstantiation is set to "true" or if the software-component defines NVRAM access via permanent blocks.			
Base	ARObject, AtpClassifier, AtpFeature, AtpStructureElement, Identifiable , Multilanguage Referrable, Referrable			
Attribute	Datatype	Mul.	Kind	Note
initValue	String	0..1	attr	Specifies initial value(s) of the PerInstanceMemory
swDataDefProps	SwDataDefProps	0..1	aggr	This represents the ability to allocate RAM at specific memory sections, for example, to support the RAM Block recovery strategy by mapping to uninitialized RAM.
type	CIdentifier	1	ref	The name of the "C"-type
typeDefinition	String	1	attr	A definition of the type with the syntax of a 'C' typedef.

Table 7.48: PerInstanceMemory

7.7.2 PerInstanceMemory typed by AUTOSAR Data Types

[TPS_SWCT_01365] [PerInstanceMemory](#) typed by AUTOSAR Data Types [A [PerInstanceMemory](#) typed with AUTOSAR data types is defined by a [VariableDataPrototype](#) in the role [arTypedPerInstanceMemory](#).

[VariableDataPrototype](#) is derived from [DataPrototype](#) which has an association to an [AutosarDataType](#).]()

This defines the data type of the AUTOSAR-typed [PerInstanceMemory](#).

[TPS_SWCT_01366] Initial value of a [PerInstanceMemory](#) typed by AUTOSAR Data Types [The [initValue](#) is described with a [ValueSpecification](#)]()

[TPS_SWCT_01367] Typed by AUTOSAR data type vs. typed by C data type [In difference to the "C" typed [PerInstanceMemory](#) the AUTOSAR-typed [PerInstanceMemory](#) is able to define information controlling the visibility in a MCD system via a [SwDataDefProps](#) for the purpose of measurement (see chapter 5.4.3) or defining an input value of an axis (see chapter 5.4.5).]()

Note: Due to the use of [AutosarDataType](#) the AUTOSAR-typed [PerInstanceMemory](#) can not support C++ specific types or pointer types directly.

7.8 Static Memory and Constant Memory

[TPS_SWCT_01368] Describe static and constant memory [Static memory (formalized by means of `InternalBehavior.staticMemory`) and constant memory (formalized by means of `InternalBehavior.constantMemory`) can be used whenever `AutosarDataTypes` should be used in the implementation of an `AtomicSwComponentType` but no involvement of the RTE (for memory allocation and management) is required.]()

This includes special cases of measurement and calibration but also debugging.

[TPS_SWCT_01483] Use static and constant memory to support Measurement and Calibration [The information about these characteristic values and variables is given with the purpose to support Measurement and Calibration (see chapter 2.2) and has to be taken into account for the generation of A2L files.

A proprietary generator shall take care of these data for the purpose of generating A2L.]()

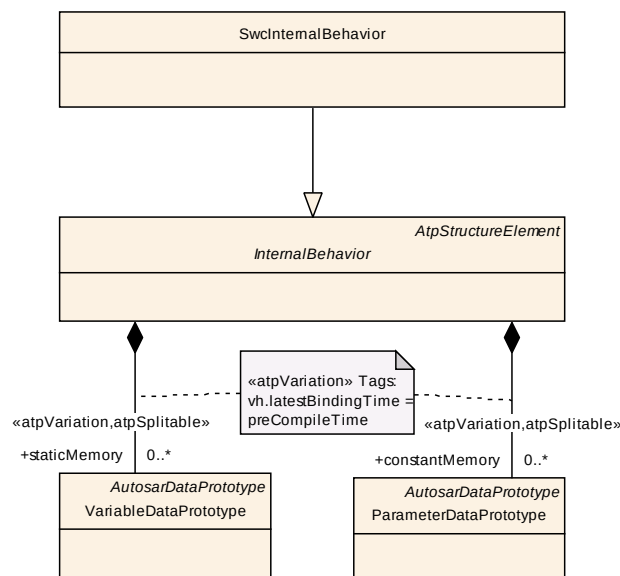


Figure 7.33: Static Memory and Constant Memory

[TPS_SWCT_01369] Static and constant memory is not instantiated by the RTE [In contrast to the other kinds of memory like `implicitInterRunnableVariable`, `implicitInterRunnableVariable`, `PerInstanceMemory`, `sharedParameter` or `perInstanceParameter` the `staticMemory` and `constantMemory` are **not** instantiated by the RTE.]()

This allows for more efficient implementations (especially for software-components provided as object code) by avoidance of the additional indirection caused by the RTE's component data structure.

Further on, this kind of memory reduces the dependencies of the software-component implementation to generated RTE code which is appreciated for safety related functionalities.

Due to the instantiation of the memory by the software-component's implementation the `constantMemory` behaves like a `sharedParameter` (see chapter 2.2.3.2)

[constr_2028] `staticMemory` is restricted to single instantiation [The `staticMemory` is only supported if the attribute `supportsMultipleInstantiation` of the owning `SwcInternalBehavior` is set to `false`]()

This constraint prevents hidden communication between `SwComponentPrototypes` of the same `SwComponentType`.

[constr_2029] `shortName` of `constantMemory` and `staticMemory` [The `shortName` of a `VariableDataPrototype` in role `staticMemory` or a `ParameterDataPrototype` in role `constantMemory` has to be equal with the 'C' identifier of the described variable resp. constant.]()

7.9 Included AUTOSAR Data Types

[TPS_SWCT_01155] `IncludedDataSet` [An `IncludedDataSet` declares that a set of `AutosarDataTypes` are used for the C / C++ implementation of the software component. The `AutosarDataTypes` become part of the contract.]()

[TPS_SWCT_01156] Required if the `AutosarDataType` is not used for any `DataPrototype` [This information is required if the `AutosarDataType` is not used for any `DataPrototype` owned by this software component or if a prefix for C language identifiers belonging to `AutosarDataTypes` shall be defined.]()

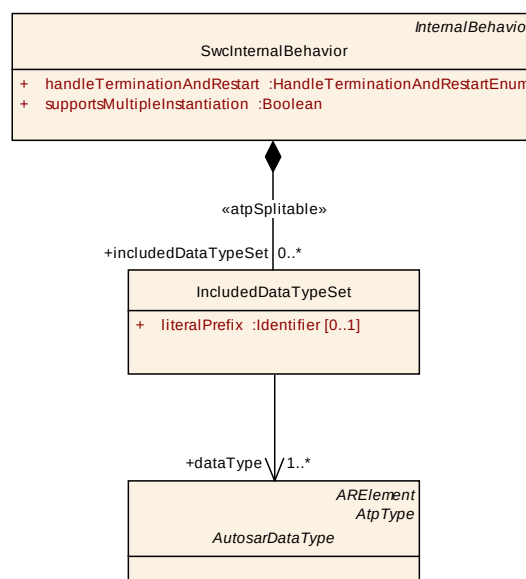


Figure 7.34: Included AUTOSAR Data Types

Class	IncludedDataTypeSet			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::IncludedDataTypes			
Note	An includedDataTypeSet declares that a set of AutosarDataType is used by a basic software module or a software component for its implementation and the AutosarDataType becomes part of the contract. This information is required if the AutosarDataType is not used for any DataPrototype owned by this software component or if the enumeration literals, lowerLimit and upperLimit constants shall be generated with a literalPrefix. The optional literalPrefix is used to add a common prefix on enumeration literals, lowerLimit and upperLimit constants created by the RTE.			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
dataType	AutosarDataTypee	1..*	ref	AutosarDataType belonging to the includedDataTypeSet
literalPrefix	Identifier	0..1	ref	LiteralPrefix defines a common prefix for all AutosarDataTypes of the includedDataTypeSet to be added on enumeration literals, lowerLimit and upperLimit constants created by the RTE.

Table 7.49: IncludedDataTypeSet

This supports the common usage of the AUTOSAR data type system for RTE provided memory objects and memory objects declared by the software component implementation.

Further on, this enables the generation of the RTE Application Types Header File for AUTOSAR services containing the required data types for the C-API before the data type usage in dedicated ports for an ECU is known.

[TPS_SWCT_01157] Attribute [literalPrefix](#) of [IncludedDataTypeSet](#) [In addition the [literalPrefix](#) might be used to separate the namespace of C language identifiers belonging to equally named [AutosarDataTypes](#) used for the same software component C implementation.]()

7.10 Included Mode Declaration Groups

[TPS_SWCT_01153] [IncludedModeDeclarationGroupSet](#) [Similar to the consideration of data types using [IncludedDataTypeSet](#), [SwcInternalBehavior](#) aggregates [IncludedModeDeclarationGroupSet](#) that in turn allows for referencing [ModeDeclarationGroups](#) with the intent to express that the referenced [ModeDeclarationGroups](#) are used in the context of the enclosing [AtomicSwComponentType](#).]()

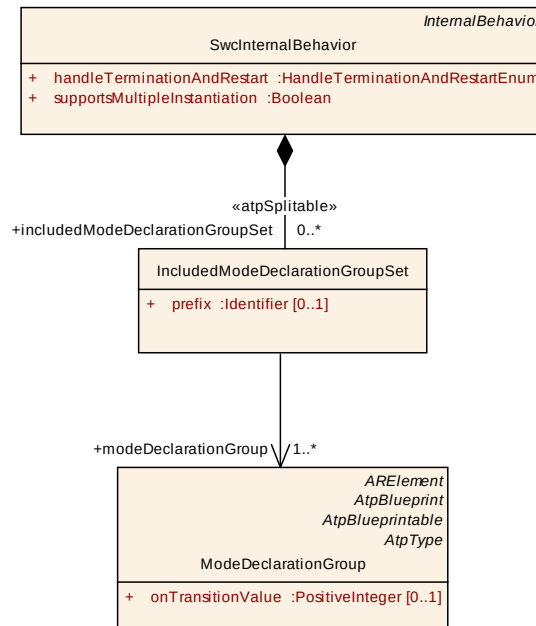


Figure 7.35: Included ModeDeclarationGroups

Class	IncludedModeDeclarationGroupSet			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::ModeDeclarationGroup			
Note	An IncludedModeDeclarationGroupSet declares that a set of ModeDeclarationGroups used by the software component for its implementation and consequently these ModeDeclarationGroups become part of the contract.			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
modeDecl arationGro up	ModeDeclaratio nGroup	1..*	ref	This represents the referenced ModeDeclarationGroup.
prefix	Identifier	0..1	ref	The prefix shall be used by the RTE generator as a prefix for the creation of symbols related to the referenced ModeDeclarationGroups, e.g RTE_TRANSITION_<ModeDeclarationGroup>.

Table 7.50: IncludedModeDeclarationGroupSet

[TPS_SWCT_01154] Attribute `prefix` of `IncludedModeDeclarationGroupSet`

⌈ The optional attribute `prefix` of `IncludedModeDeclarationGroupSet` can be used to define a prefix that the RTE generator shall use to define symbols related to the included `ModeDeclarationGroups` with the intent to avoid potential name clashes. ⌋()

Rationale: If the attribute `prefix` is required, changes to software-component source code may be necessary.

7.11 Service Needs

7.11.1 Overview

[TPS_SWCT_01043] **ApplicationSwComponentTypes** are independent from actual ECU Hardware [[ApplicationSwComponentTypes](#) are designed to be independent of their mapping to actual ECU Hardware.]([RS_SWCT_02060](#))

However, each software-component might need services which are provided by the ECU Basic Software through AUTOSAR Services.

[TPS_SWCT_01044] **ServiceNeeds** [The [ServiceNeeds](#) (see Figures [7.36](#), [7.37](#), [7.38](#), and [7.39](#)) are used to provide detailed information what the software-component expects from the AUTOSAR Services when integrated on an actual ECU.

Note that only [AtomicSwComponentTypes](#) and [NvBlockSwComponentTypes](#) can be connected to AUTOSAR Services.]([RS_SWCT_02060](#))

[TPS_SWCT_01045] **Actual values of ECU configuration parameters fulfill the requirements given by the ServiceNeeds** [When integrating application software-components on an ECU, the actual values of ECU configuration parameters shall be chosen so that they fulfill the requirements given by the [ServiceNeeds](#) of all the integrated [AtomicSwComponentTypes](#).]([RS_SWCT_02060](#))

Note that the actual values of configuration parameters will in addition depend on the properties of the basic software and the hardware of that specific ECU, see also chapter [11](#).

For further information about the relation between the [ServiceNeeds](#) and the ECU configuration parameters see [29].

Class	ServiceNeeds (abstract)			
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
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			
Attribute	Datatype	Mul.	Kind	Note
—	—	—	—	—

Table 7.51: ServiceNeeds

The meta-class [ServiceNeeds](#) and the sub-classes for several Services are located in the `CommonStructure` package of the meta-model because they are also used in the Basic Software Module Description Template [7].

The meta-classes derived from [ServiceNeeds](#) is shown in the next three figures.

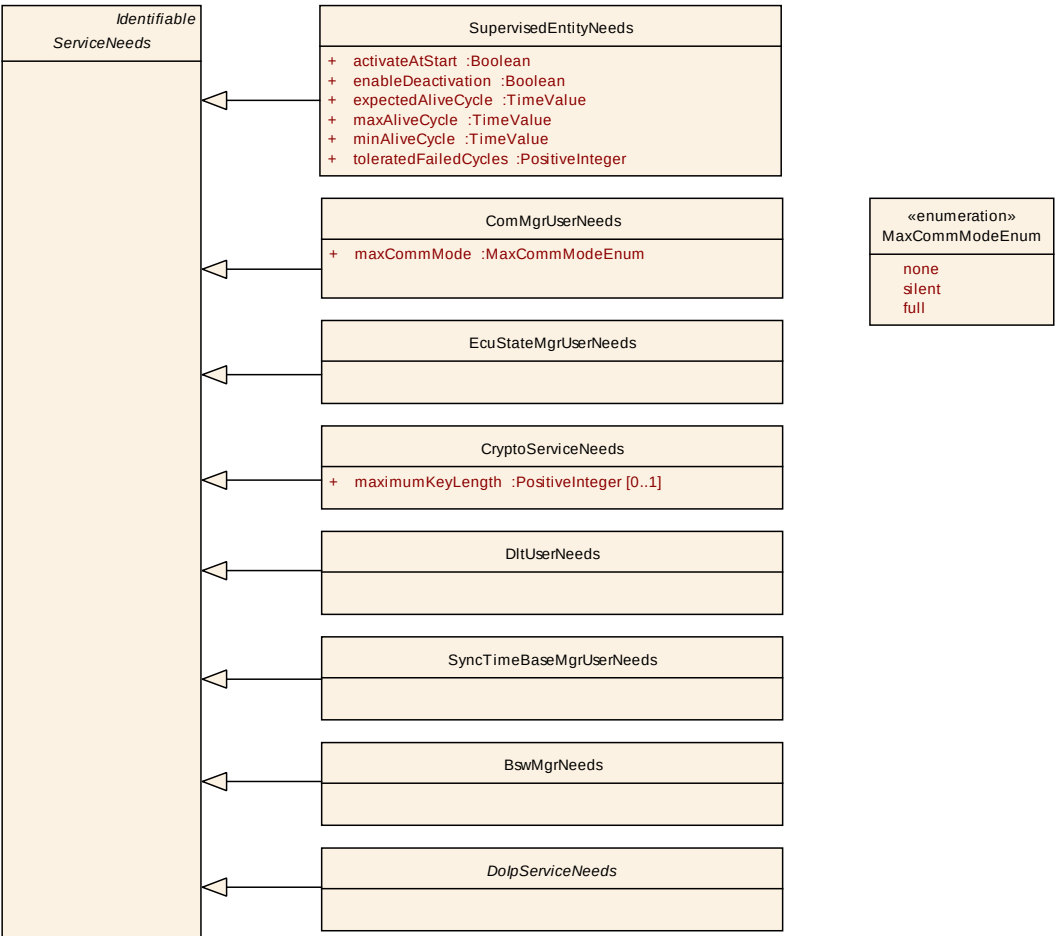


Figure 7.36: **ServiceNeeds**: General **ServiceNeeds**

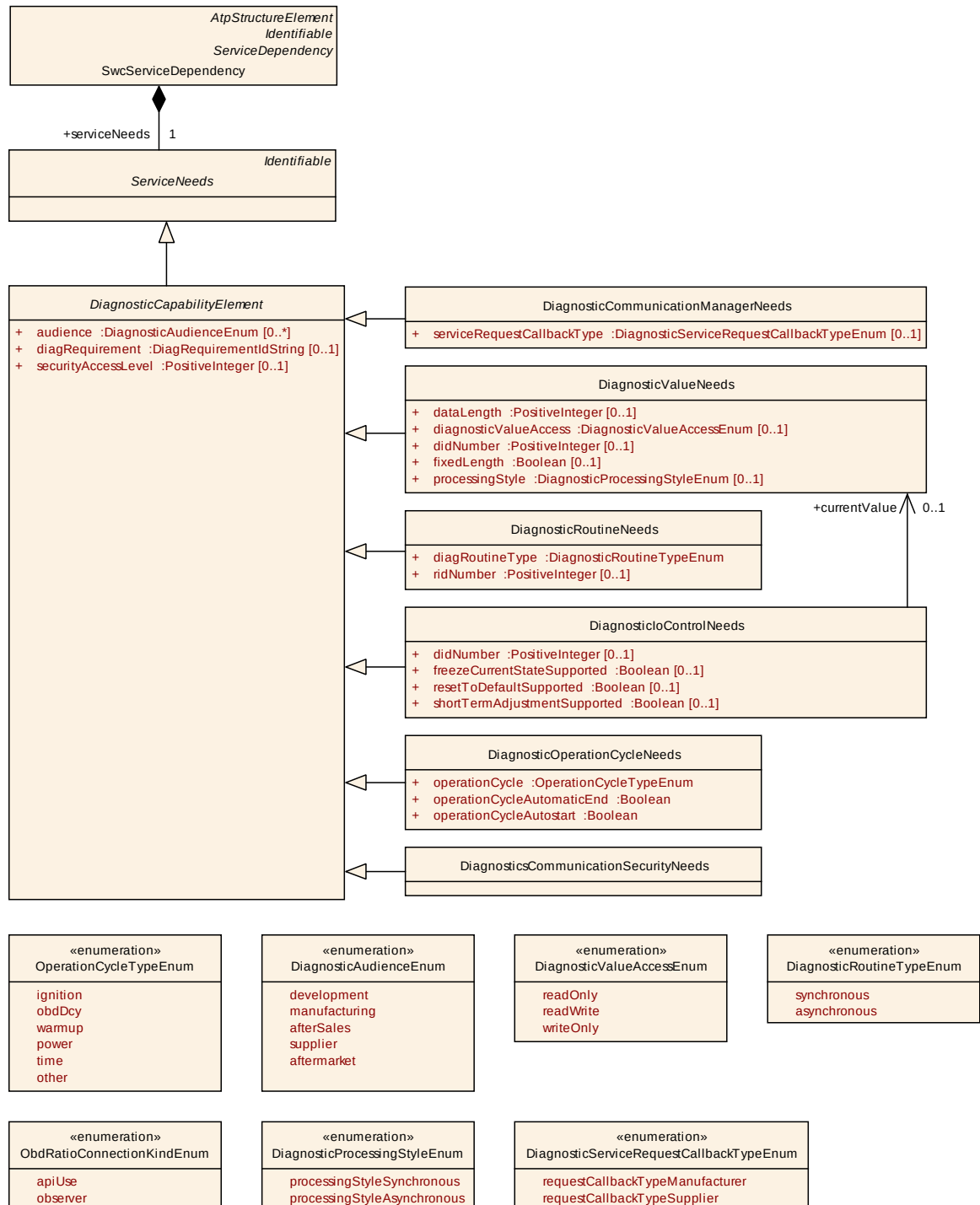


Figure 7.37: General diagnostic service-related **ServiceNeeds**

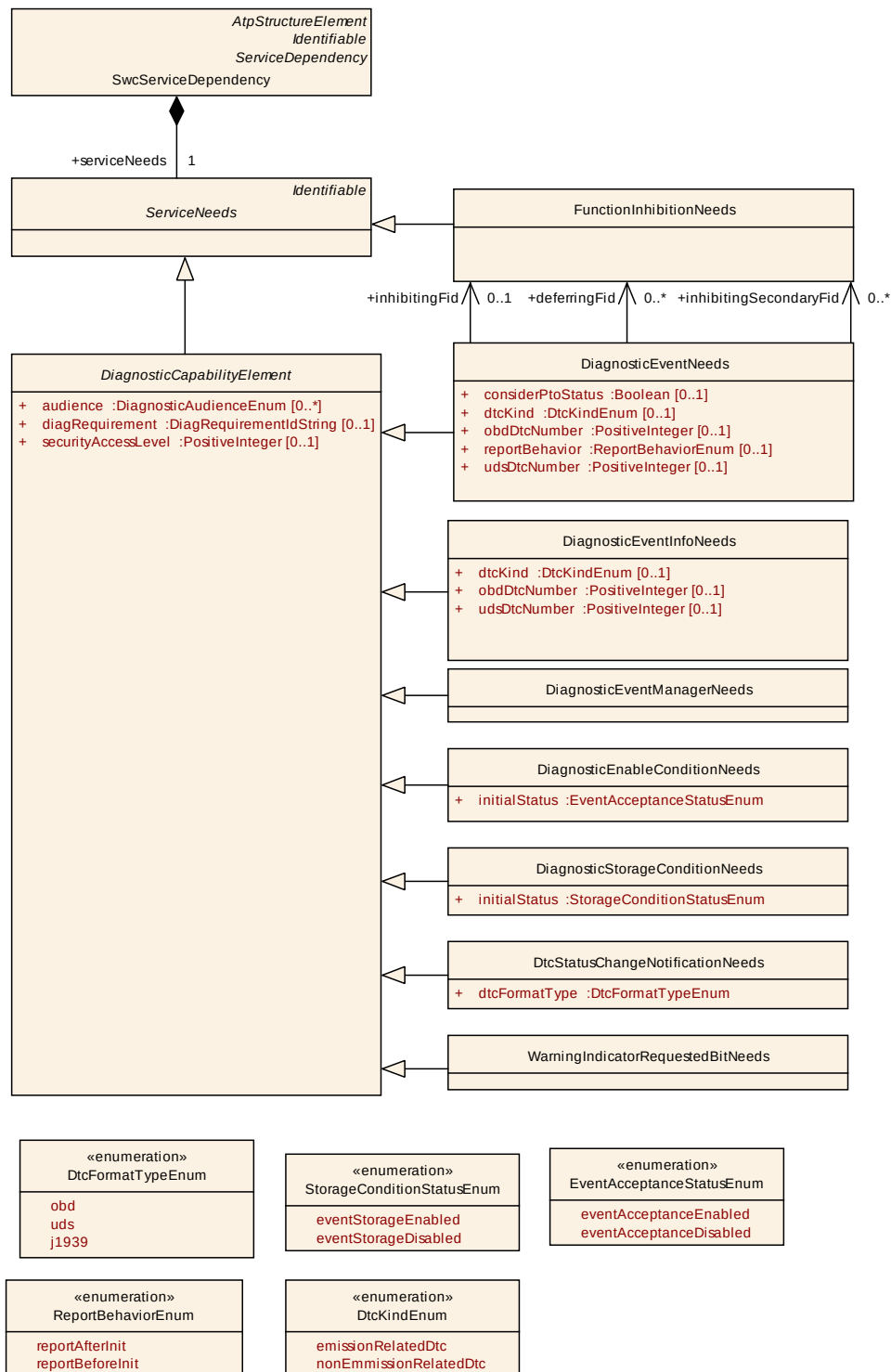


Figure 7.38: General diagnostic event-handling related **ServiceNeeds**

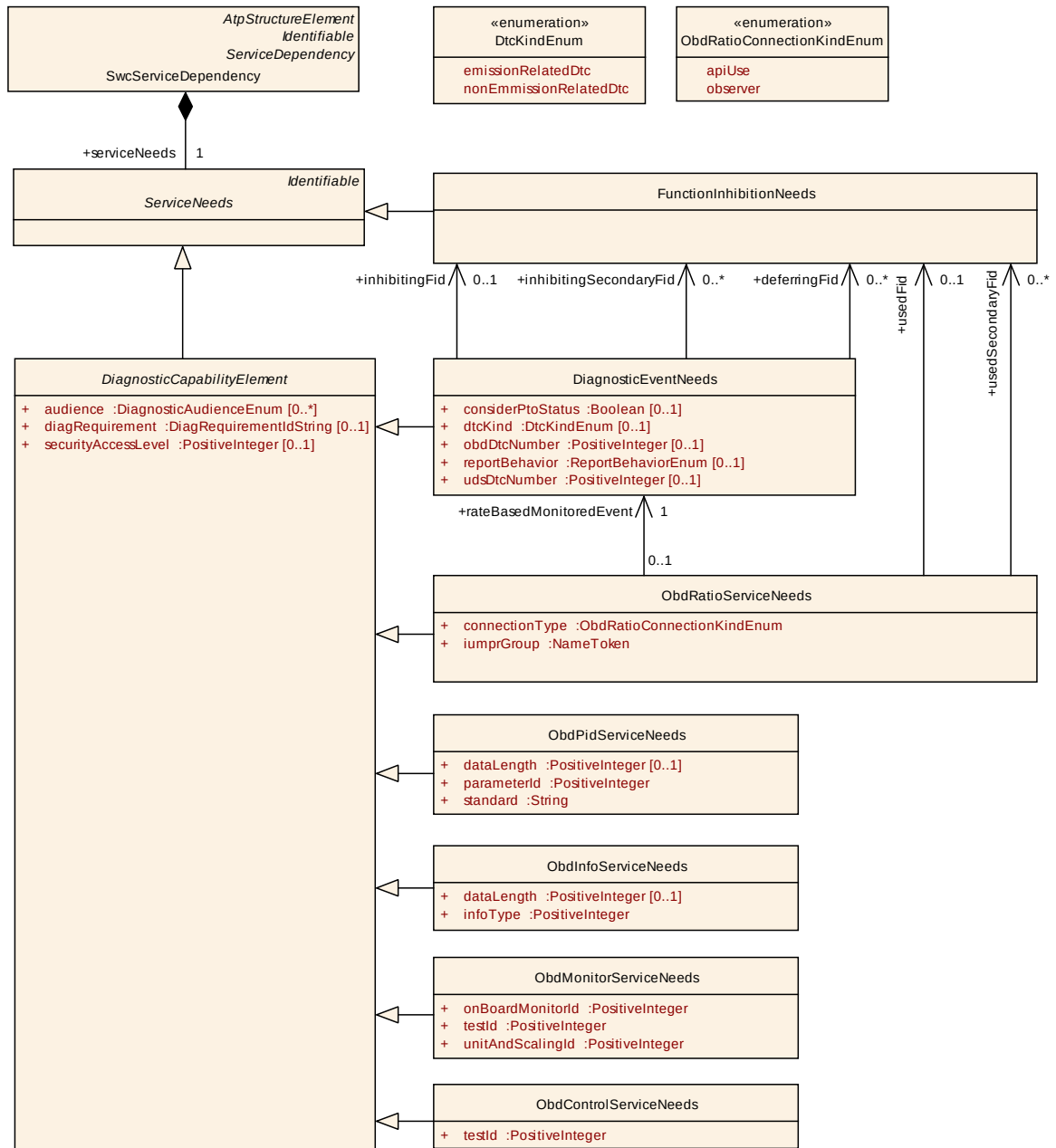


Figure 7.39: **ServiceNeeds**: Diagnostic-related **ServiceNeeds** with emphasis on OBD

7.11.2 Assignment of Service Needs to Ports and Data

[TPS_SWCT_01046] **ServiceNeeds** are defined in the scope of the **SwcInternalBehavior** [**ServiceNeeds** specified by **AtomicSwComponentTypes** are defined in the scope of the **SwcInternalBehavior** because in several cases they need associations to other parts of the **SwcInternalBehavior**.

In most cases they are related to certain **PortPrototypes** belonging to the **AtomicSwComponentTypes** because **AtomicSwComponentTypes** communicate with AUTOSAR Services via these **PortPrototypes**.](RS_SWCT_02060)

In addition, a `ServiceNeeds` element can also have relations to some data declared within the same `SwcInternalBehavior`, namely some use cases of the NVRAM Service require a `Permanent RAM Block` and/or `ROM Block` declared in the context of the single software component.

A further use case requires that a `ServiceNeeds` element is linked to a `PortGroup`. Especially, a `ServiceNeeds` can represent a group of `PortPrototypes` as input to configure the communication manager in order to handle the communication state of those `PortPrototypes`.

These relationships to `PortPrototypes`, data and `PortGroups` are required as input for tools in order to generate the XML descriptions and configurations of the basic software which implements the Service according to the needs of several `Atomic-SwComponentTypes` are integrated on an ECU, see chapter 11.

The relationship to `PortPrototypes` is defined via the meta-class `RoleBasedPortAssignment` and the relationship to data is defined via the meta-class `RoleBasedDataAssignment`.

Both are aggregating an attribute `role` which allows to defined the role of the `PortPrototypes` or data in the specific context.

[constr_2027] `SwcServiceDependency` shall be defined for service ports only [A `PortPrototype` that is referenced by a `SwcServiceDependency` via `assignedPort` shall be typed by a `PortInterface` that has `isService` set to `true`.

This rule does **not** apply to `PortPrototypes` used in the context of NV data management, i.e. for connections between an `ApplicationSwComponentType` and an `NvBlockSwComponentType`.]()

Please consider: it is permitted that a `SwcServiceDependency` containing a `DiagnosticValueNeeds` may reference via `assignedData` a `dataElement` instance in a `PPortPrototype` typed by a `SenderReceiverInterface` that has its attribute `isService` set to `false`.

The actual mapping between the `ServiceNeeds` element and its various relationships is provided by the meta-class `SwcServiceDependency` as shown in figure 7.41.

Note the difference between the associations to `PortPrototypes` and to `PortGroups`: While the `RoleBasedPortAssignment` is part of the `SwcInternalBehavior` a `PortGroup` is defined for the `SwComponentType` (thus belongs to the VFB level) and it is linked to the `PortGroups` of other `SwComponentTypes`.

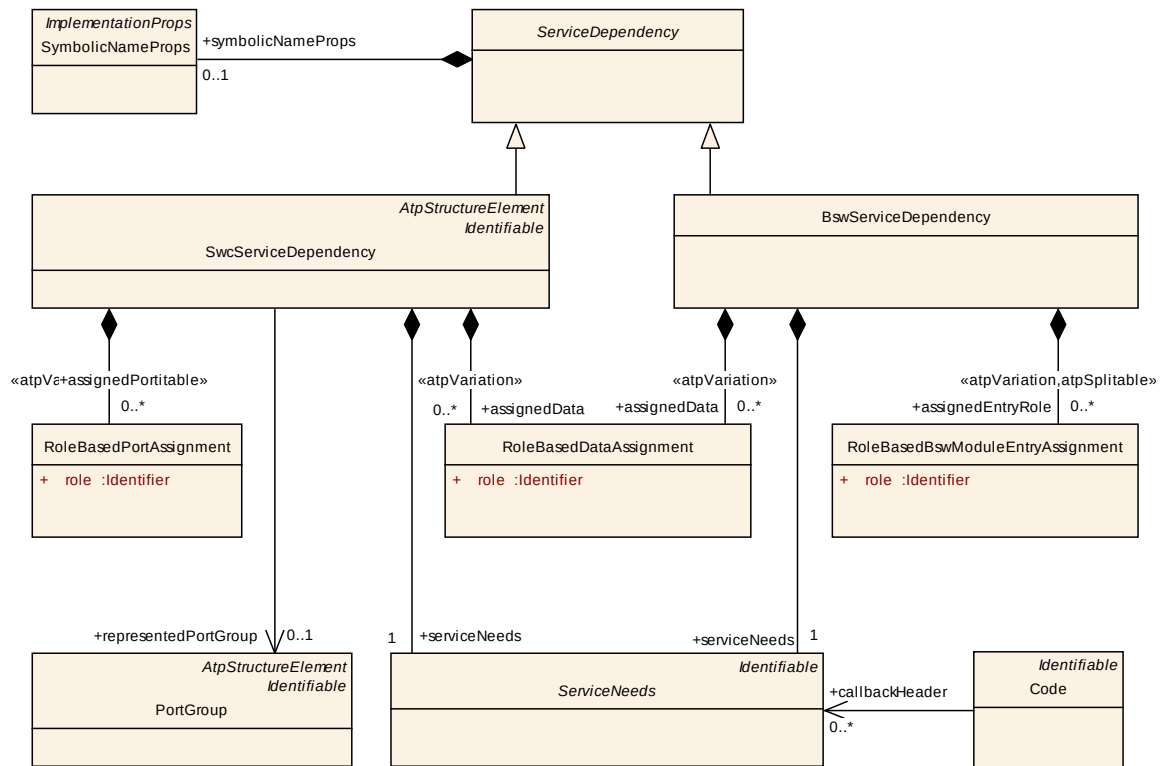


Figure 7.40: *ServiceDependency* is the abstract base class of *SwcServiceDependency*

This means a *PortGroup* represents a system feature, whereas the *RoleBasedPortAssignment* is a local feature for the purpose of communication with the AUTOSAR Service.

[TPS_SWCT_01556] Rule for setting *RoleBasedPortAssignment.role* [The value of *RoleBasedPortAssignment.role* cannot arbitrarily set but shall to equal to the *shortName* of the applicable *PortInterface* taken from the standardized AUTOSAR Service Interface model (this implies that the *category* of the *ARPackage* that owns the *PortInterface* is set to BLUEPRINT⁶ and the top-most *ARPackage.shortName* is set to AUTOSAR, see also [30]).]()

⁶see [TPS_STDT_00033]

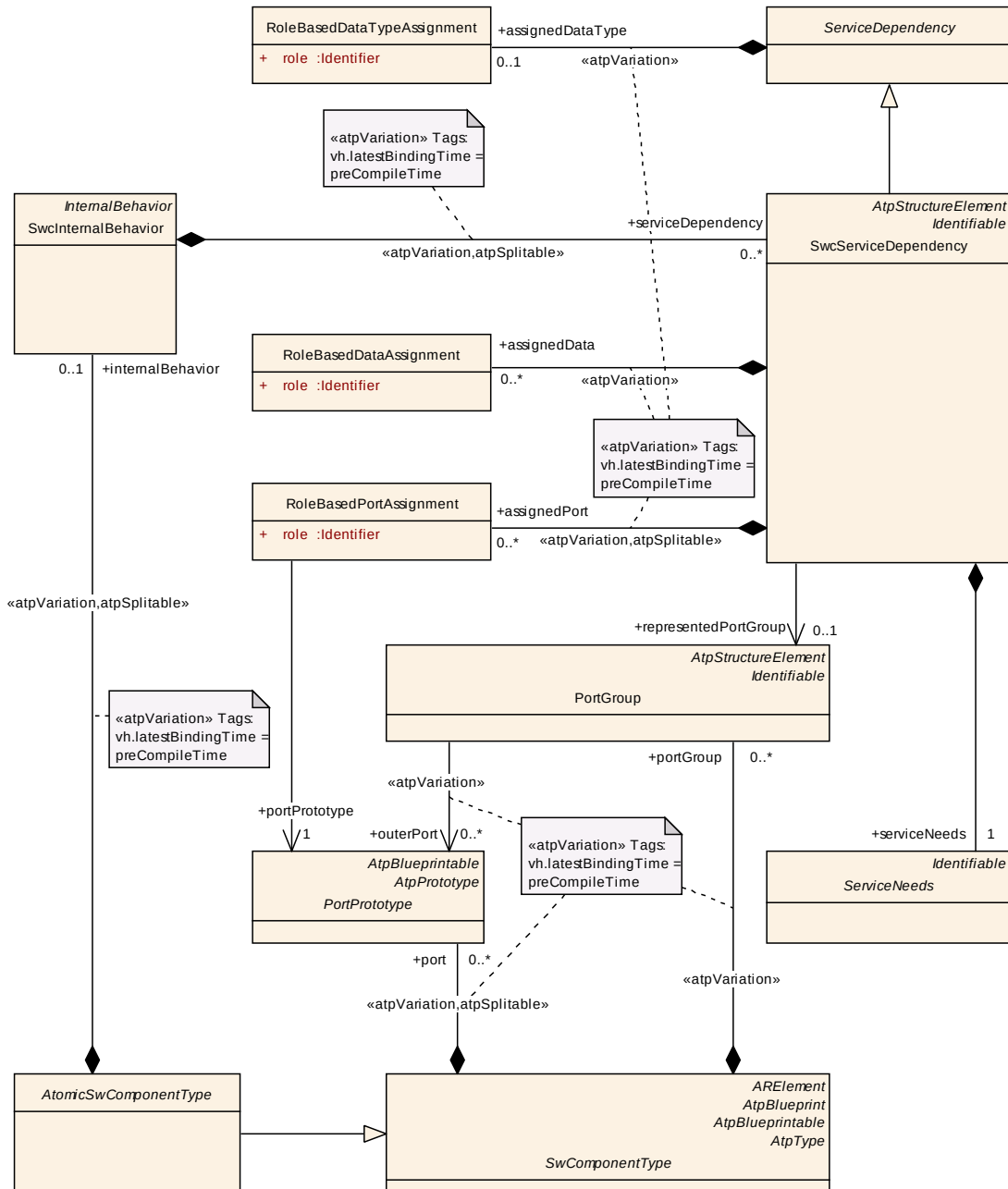


Figure 7.41: **SwcServiceDependency** in the **SwcInternalBehavior**

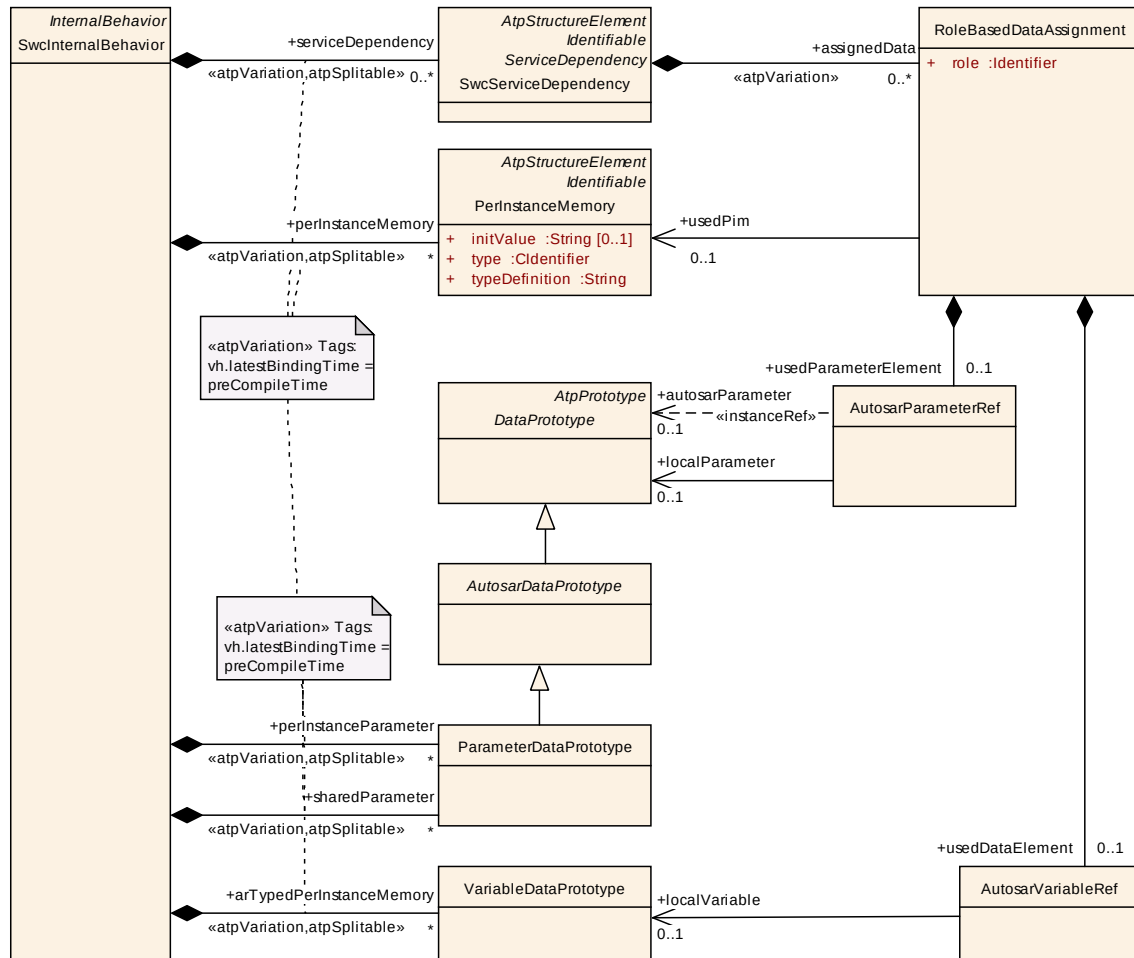


Figure 7.42: Details of **RoleBasedDataAssignment** for local data

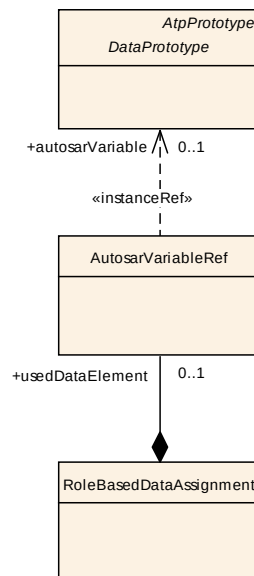


Figure 7.43: Details of **RoleBasedDataAssignment** for accessing **DataPrototypes** in **PortPrototypes**

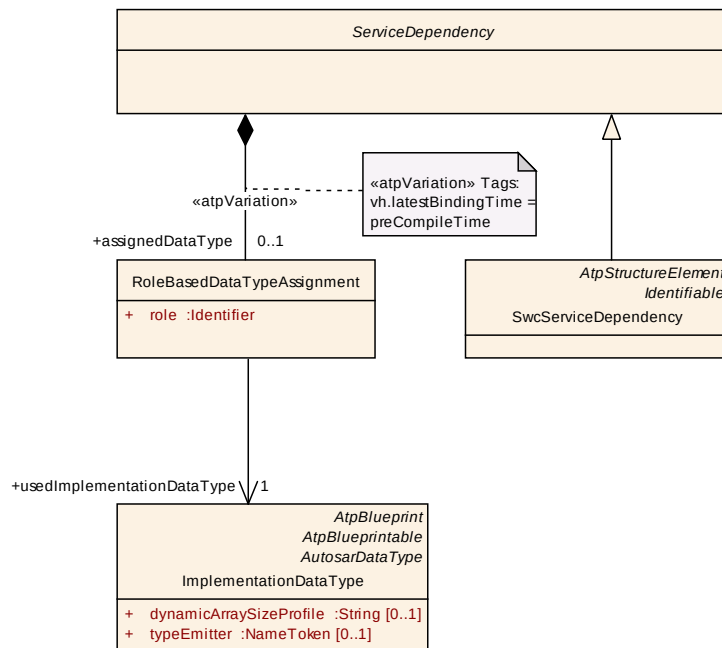


Figure 7.44: Details of **RoleBasedDataTypeAssignment** for local data

Class	ServiceDependency (abstract)			
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
Note	Collects all dependencies of a software module or component on an AUTOSAR Service related to a specific item (e.g. an NVRAM Block, a diagnostic event etc.). It defines the quality of service (ServiceNeeds) of this item as well as (optionally) references to additional elements. This information is required for tools in order to generate the related basic software configuration and ServiceSwComponentTypes.			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
assignedD ataType	RoleBasedData TypeAssignmen t	0..1	aggr	This is the role of the assignment data type in the given context. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime
symbolicN ameProps	SymbolicName Props	0..1	aggr	This attribute can be taken to contribute to the creation of symbolic name values.

Table 7.52: ServiceDependency

Class	SwcServiceDependency			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::Service Mapping			
Note	Specialization of ServiceDependency in the context of an SwcInternalBehavior. It allows to associate ports, port groups and (in special cases) data defined for an atomic software component to a given ServiceNeeds element.			
Base	ARObject, AtpClassifier, AtpFeature, AtpStructureElement, Identifiable , Multilanguage Referrable, Referrable , ServiceDependency			
Attribute	Datatype	Mul.	Kind	Note
assignedData	RoleBasedData Assignment	*	aggr	Defines the role of an associated data object of the same component. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime
assignedPort	RoleBasedPort Assignment	*	aggr	Defines the role of an associated port of the same component. Stereotypes: atpSplittable; atpVariation Tags: atp.Splitkey=assignedPort, variation Point.shortLabel vh.latestBindingTime=preCompileTime
representedPortGroup	PortGroup	0..1	ref	This reference specifies an association between the ServiceNeeds and a PortGroup, for example to request a communication mode which applies for communication via these ports. The referred PortGroup shall be local to this atomic SWC, but via the links between the PortGroups, a tool can evaluate this information such that all the ports linked via this port group on the same ECU can be found.
serviceNeeds	ServiceNeeds	1	aggr	The associated ServiceNeeds.

Table 7.53: SwcServiceDependency

Class	SymbolicNameProps			
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
Note	This meta-class can be taken to contribute to the creation of symbolic name values.			
Base	ARObject, ImplementationProps , Referrable			
Attribute	Datatype	Mul.	Kind	Note
—	—	—	—	—

Table 7.54: SymbolicNameProps

Class	RoleBasedPortAssignment			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::Service Mapping			
Note	This class specifies an assignment of a role to a particular service port (RPortPrototype or PPortPrototype) of an AtomicSwComponentType. With this assignment, the role of the service port can be mapped to a specific ServiceNeeds element, so that a tool is able to create the correct connector.			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
portPrototype	PortPrototype	1	ref	Service port used in the assigned role. This port shall either belong to the same AtomicSoftwareComponent as the SwcInternalBehavior which owns the ServiceDependency or to the same NvBlockComponentType as the NvBlockDescriptor.
role	Identifier	1	ref	This is the role of the assigned Port in the given context. The value shall be a shortName of the Blueprint of a PortInterface as standardized in the Software Specification of the related AUTOSAR Service.

Table 7.55: RoleBasedPortAssignment

Class	RoleBasedDataAssignment			
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
Note	This class specifies an assignment of a role to a particular data object in the SwcInternalBehavior of a software component (or in the BswModuleBehavior of a module or cluster) in the context of an AUTOSAR Service. With this assignment, the role of the data can be mapped to a specific ServiceNeeds element, so that a tool is able to create the correct access.			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note

Attribute	Datatype	Mul.	Kind	Note
role	Identifier	1	ref	This is the role of the assigned data in the given context, for example for an NVRAM Block it is used to distinguish between an mirror block and a ROM default block. Possible values need to be specified on M1 level. This also is intended to support the so called "Signal based Approach" of the DCM. In this use case the name of the involved data element is required. This name shall be taken from the DataElement referenced by the property usedDataElement. The following values are standardized: • ramBlock indicates data to be used as a mirror for an NVRAM Block. • defaultData indicates constant data to be used as default in the context of this ServiceNeeds, e.g. for an NVRAM Block. • signalBasedDiagnostics indicates the RoleBasedDataAssignment shall be used for signal based diagnostics.
usedDataElement	AutosarVariableRef	0..1	aggr	The VariableDataPrototype used in this role, e.g. • Permanent RAM Block of an NVRAM Block which shall belong to the same SwcInternalBehavior or BswInternalBehavior. • In the role signalBasedDiagnostics it has to refer to a VariableDataPrototype in a SenderReceiverInterface or a NvDataInterface.
usedParameterElement	AutosarParameterRef	0..1	aggr	The ParameterDataPrototype used in this role, e.g. • ROM Block of an NVRAM Block. It shall belong to the same SwcInternalBehavior or BswInternalbehavior. • In the role signalBasedDiagnostics it has to refer to a ParameterDataPrototype in a ParameterInterface.
usedPim	PerInstanceMemory	0..1	ref	The (untyped) PerInstanceMemory used in this role (e.g. as a Permanent RAM Block for an NVRAM Block).

Table 7.56: RoleBasedDataAssignment

Class	RoleBasedDataTypeAssignment			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::ServiceMapping			
Note	This class specifies an assignment of a role to a particular data type of a software component (or in the BswModuleBehavior of a module or cluster) in the context of an AUTOSAR Service. With this assignment, the role of the data type can be mapped to a specific ServiceNeeds element, so that a tool is able to create the correct access.			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
role	Identifier	1	ref	This is the role of the associated data type in the given context.
usedImplementationDataType	ImplementationDataType	1	ref	This represents the associated ImplementationDataType.

Table 7.57: RoleBasedDataTypeAssignment

7.11.3 Specific Service Dependencies

7.11.3.1 NvM Service Dependencies

This chapter describes the usage of the specific meta-classes derived from [ServiceNeeds](#) within an [AtomicSwComponentType](#).

The meta-class [NvBlockNeeds](#) is used to define requirements to configure the NVRAM Manager Service. In addition, it may define requirements how the RTE shall implement writing strategies of an [NvBlockSwComponentType](#).

An [SwcInternalBehavior](#) may provide several [SwcServiceDependency](#)s that in turn aggregate an [NvBlockNeeds](#) element where each defines the requirements from one NVRAM Block (for more information on the AUTOSAR NVRAM Manager see [31]).

There are several use cases how a software-component can interact with the NVRAM Manager service. Each use case is discussed in a separate sub-chapter.

Class	NvBlockNeeds			
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
Note	Specifies the abstract needs on the configuration of a single NVRAM Block.			
Base	ARObject, Identifiable , MultilanguageReferrable, Referrable , ServiceNeeds			
Attribute	Datatype	Mul.	Kind	Note
calcRamBlockCrc	Boolean	0..1	attr	Defines if CRC (re)calculation for the permanent RAM Block is required.
checkStaticBlockId	Boolean	0..1	attr	Defines if the Static Block Id check shall be enabled.
cyclicWritingPeriod	TimeValue	0..1	attr	This represents the period for cyclic writing of NvData to store the associated RAM Block.

Attribute	Datatype	Mul.	Kind	Note
nDataSets	PositiveInteger	0..1	attr	Number of data sets to be provided by the NVRAM manager for this block. This is the total number of ROM Blocks and RAM Blocks.
nRomBlocks	PositiveInteger	0..1	attr	Number of ROM Blocks to be provided by the NVRAM manager for this block. Please note that these multiple ROM Blocks are given in a contiguous area.
ramBlockStatusControl	RamBlockStatusControlEnum	0..1	attr	This attribute defines how the management of the RAM Block status is controlled.
readonly	Boolean	0..1	attr	True: data of this NVRAM Block are write protected for normal operation (but protection can be disabled) false: no restriction
reliability	NvBlockNeedsReliabilityEnum	0..1	attr	Reliability against data loss on the non-volatile medium.
resistantToChangedSW	Boolean	0..1	attr	Defines whether an NVRAM Block shall be treated resistant to configuration changes (true) or not (false). For details how to handle initialization in the latter case, please refer to the NVRAM specification.
restoreAtStart	Boolean	0..1	attr	Defines whether the associated RAM Block shall be implicitly restored during startup by the basic software.
storeAtShutdown	Boolean	0..1	attr	Defines whether or not the associated RAM Block shall be implicitly stored during shutdown by the basic software.
storeCyclic	Boolean	0..1	attr	Defines whether or not the associated RAM Block shall be implicitly stored periodically by the basic software.
storeEmergency	Boolean	0..1	attr	Defines whether or not the associated RAM Block shall be implicitly stored in case of ECU failure (e.g. loss of power) by the basic software. If the attribute storeEmergency is set to true the associated RAM Block shall be configured to have immediate priority.
storeImmediate	Boolean	0..1	attr	Defines whether or not the associated RAM Block shall be implicitly stored immediately during or after execution of the according SW-C RunnableEntity by the basic software.
useAutoValidationAtShutdown	Boolean	0..1	attr	If set to true the RAM Block shall be auto validated during shutdown phase.
useCRCComparisonMechanism	Boolean	0..1	attr	If set to true the CRC of the RAM Block shall be compared during a write job with the CRC which was calculated during the last successful read or write job in order to skip unnecessary NVRAM writings.
writeOnlyOnce	Boolean	0..1	attr	Defines write protection after first write: true: This block is prevented from being changed/erased or being replaced with the default ROM data after first initialization by the software-component. false: No such restriction.

Attribute	Datatype	Mul.	Kind	Note
writeVerification	Boolean	0..1	attr	Defines if Write Verification shall be enabled for this NVRAM Block.
writingFrequency	PositiveInteger	0..1	attr	Provides the amount of updates to this block from the application point of view. It has to be provided in "number of write access per year".
writingPriority	NvBlockNeedsWritingPriorityEnum	0..1	attr	Requires the priority of writing this block in case of concurrent requests to write other blocks.

Table 7.58: NvBlockNeeds

Enumeration	RamBlockStatusControlEnum
Package	M2::AUTOSARTemplates::SWComponentTemplate::NvBlockComponent
Note	This enumeration type defines options for how the management of the ramBlock status is controlled.
Literal	Description
api	The ramBlock status is controlled via service interface by usage of the SetRamBlockStatus operation.
nvRamManager	The ramBlock status is controlled exclusively by the Nv Ram Manager.

Table 7.59: RamBlockStatusControlEnum

7.11.3.1.1 Nvm Use Case: Permanent RAM Block

Scenario: a [AtomicSwComponentType](#) is using an an NVRAM Block with a Permanent RAM Block implemented by a [PerInstanceMemory](#) section or a [VariableDataPrototype](#) in the role [arTypedPerInstanceMemory](#). In either case, the required memory for the Permanent RAM Block is allocated by the RTE during ECU Configuration.

In this case the following rules apply:

[TPS_SWCT_02501] Setup for Nvm Use Case: Permanent RAM Block [

RoleBasedPortAssignment

For every used [ClientServerInterface](#) provided by the NvM it is necessary to create a [RoleBasedPortAssignment](#) and set the value of the attribute [role](#) of the [RoleBasedPortAssignment](#) to the name of the used standardized [ClientServerInterface](#). The following [ClientServerInterfaces](#) shall (i.e. lower multiplicity > 0) or can (lower multiplicity = 0) be used in this context:

- NvmService [0 .. 1]
- NvMNotifyJobFinished [0 .. 1]
- NvMNotifyInitBlock [0 .. 1]

- NvMAdmin [0 .. 1]

RoleBasedDataAssignment

`RoleBasedDataAssignment` shall be created that refers to either the `PerInstanceMemory` in the role `usedPim` or to the `VariableDataPrototype` in the role `usedDataElement`. The value of the attribute `role` of the `RoleBasedDataAssignment` shall be set to `ramBlock`.

Optionally, it is possible to create an additional `RoleBasedDataAssignment` to a `ParameterDataPrototype` in the role `usedParameterElement`. The value of the `ParameterDataPrototype` is then taken as the initial or default value for the NVRAM Block. In this case the value of the attribute `role` of the `RoleBasedDataAssignment` shall be set to `defaultValue`. Therefore, the following roles are applicable:

- `ramBlock` [1]
- `defaultValue` [0 .. 1]

RepresentedPortGroup

N/A

]()

For more information please refer to [SWS_NvM_00734], [SWS_NvM_00735], [SWS_NvM_00736], and [SWS_NvM_00737].

The same mechanism (see description of scenario) applies also for an `NvBlock-SwComponentType`. For each NVRAM Block the NVRAM Manager can be configured (with the help of `SwcServiceDependency.assignedData`) to use the same Permanent RAM Block.

It is the responsibility of the NVRAM Manager to provide the content of the NVRAM Block in this Permanent RAM Block during startup or on explicit request and to write back the content to the storage medium during shut-down or on explicit request.

7.11.3.1.2 Nvm Use Case: Temporary RAM Block

Scenario: an `AtomicSwComponentType` is using some NVRAM Block with a Temporary RAM Block.

In this case the `AtomicSwComponentType` is responsible for allocating the allocation of sufficient memory. In other words, the `AtomicSwComponentType` shall provide a memory area that is available to the API call to the NVRAM Manager for storage of the NV data.

[TPS_SWCT_02502] Setup for Nvm Use Case: Temporary RAM Block [

RoleBasedPortAssignment

This is mandatory for the described scenario. For every used `ClientServer-`

Interface provided by the `Nvm` it is necessary to create a `RoleBasedPortAssignment` and set the value of the attribute `role` of the `RoleBasedPortAssignment` to the name of the used `ClientServerInterface`. The following `ClientServerInterfaces` shall (i.e. lower multiplicity > 0) or can (lower multiplicity = 0) be used in this context:

- `NvmService` [1]
- `NvMNotifyJobFinished` [0 .. 1]
- `NvMNotifyInitBlock` [0 .. 1]
- `NvMAdmin` [0 .. 1]

RoleBasedDataAssignment

The usage of a `RoleBasedDataAssignment` with attribute `role` set to `defaultValue` is optional and depends on whether or not an initial value is required.

- `defaultValue` [0..1]

RoleBasedDataTypeAssignment

By this means it is possible to define the data type of a Temporary RAM Block. The data type information can be used to calculate the NVRAM Block size. [constr_1301] applies.

- `temporaryRamBlock` [0..1]

RepresentedPortGroup

N/A

]()

[constr_1301] Existence of `RoleBasedDataTypeAssignment.role` vs. `RoleBasedDataAssignment.role` [The usage of a `RoleBasedDataTypeAssignment` with attribute `role` set to the value `temporaryRamBlock` is only allowed if **no** `RoleBasedDataAssignment` defined with attribute `role` set to value `defaultValue` exists in the owning `SwcServiceDependency`.]()

The rationale for [constr_1301] is that the existence of a `RoleBasedDataAssignment` would already provide sufficient information for the intended purpose. The parallel existence of a `RoleBasedDataTypeAssignment` is therefore fully redundant and could only lead to potential inconsistencies.

For more information please refer to [SWS_NvM_00734], [SWS_NvM_00735], [SWS_NvM_00736], and [SWS_NvM_00737].

7.11.3.1.3 Nvm Use Case: RAM Block with explicit synchronization using Mirror Interfaces

Scenario: an `AtomicSwComponentType` is using an NVRAM Block where the RAM Block uses explicit synchronization by means of mirror interfaces. In this case the RAM Block does not necessarily have to be formally described by means of a `PerInstanceMemory` or a `VariableDataPrototype` in the role `arTypedPerInstanceMemory`.

Consequently, the software-component itself is responsible for the allocation of memory. On the other hand, this can also mean that the software-component can use several RAM Blocks instead of just one RAM Block.

[TPS_SWCT_02504] Setup for Nvm Use Case: RAM Block with explicit synchronization using Mirror Interfaces [

RoleBasedPortAssignment

This is mandatory for the described scenario. For every used `ClientServerInterface` provided by the Nvm it is necessary to create a `RoleBasedPortAssignment` and set the value of the attribute `role` of the `RoleBasedPortAssignment` to the name of the used `ClientServerInterface`. The following `ClientServerInterfaces` shall (i.e. lower multiplicity > 0) or can (lower multiplicity = 0) be used in this context:

- `NvMService` [0..1]
- `NvMNotifyJobFinished` [0..1]
- `NvMNotifyInitBlock` [0..1]
- `NvMAdmin` [0..1]
- `NvMMirror` [1]

RoleBasedDataAssignment

In this scenario the existence of a `RoleBasedDataAssignment` is optional. The `RoleBasedDataAssignment` needs to reference a `ParameterDataPrototype` aggregated by the enclosing `SwcInternalBehavior` in the role `perInstanceParameter` or `sharedParameter`.

- `defaultValue` [0..1]

RoleBasedDataTypeAssignment

By this means it is possible to define the data type of a temporary RAM Block and used internal data structure in case of explicit synchronization with `NvMMirror` interface respectively. The data type information can be used to calculate the NVRAM Block size and minimum Permanent RAM Block size. [\[constr_1301\]](#) applies.

- `temporaryRamBlock` [0..1]

RepresentedPortGroup

N/A

]()

For more information please refer to [SWS_NvM_00734], [SWS_NvM_00735], [SWS_NvM_00736], [SWS_NvM_00737], and [SWS_NvM_00738].

7.11.3.1.4 NVM Use Case: Software-Components using Nv Data provided by NvBlockSwComponentType (not ServiceSwComponent of NvM)

Scenario: an [AtomicSwComponentType](#) is using an NVRAM Block managed by an [NvBlockSwComponentType](#) (see section 11.5.2, as opposed to an NVRAM Block provided by a [ServiceSwComponentType](#)). Constraints [[constr_1148](#)], [[constr_1149](#)], and [[constr_2011](#)] apply.

[TPS_SWCT_02503] Setup for NVM Use Case: Software-Components using Nv Data provided by NvBlockSwComponentType [

RoleBasedPortAssignment

This is mandatory for the described scenario. For every used [ClientServerInterface](#) provided by the NvM it is necessary to create a [RoleBasedPortAssignment](#) and set the value of the attribute [role](#) of the [RoleBasedPortAssignment](#) to the name of the used [ClientServerInterface](#). The following [ClientServerInterfaces](#) shall (i.e. lower multiplicity > 0) or can (lower multiplicity = 0) be used in this context:

- NvMService [0..1]
- NvMNotifyJobFinished [0..1]
- NvMNotifyInitBlock [0..1]
- NvMAdmin [0..1]

For every [PortPrototype](#) of a software-component typed by an [NvDataInterface](#) defining a [SwcServiceDependency](#) it is necessary to create a [RoleBasedPortAssignment](#) and set the value of the attribute [role](#) of the attribute [assignedPort](#) to the value `NvDataPort`:

- NvDataPort [1..*]

RoleBasedDataAssignment

N/A

RepresentedPortGroup

N/A

]()

For more information please refer to [SWS_NvM_00734], [SWS_NvM_00735], [SWS_NvM_00736], and [SWS_NvM_00737]. Note that [NvBlockNeeds](#) described in Chapter 11.5.4) is not in the scope of this use case.

7.11.3.2 Watchdog Service Dependencies

The meta-class [SupervisedEntityNeeds](#) is used to define requirements to configure the Watchdog Service. For the terms related to the AUTOSAR Watchdog Manager see [32].

7.11.3.2.1 Watchdog Service use Case: Supervision

Class	SupervisedEntityNeeds			
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
Note	Specifies the abstract needs on the configuration of the Watchdog Manager for one specific Supervised Entity.			
Base	ARObject, Identifiable , MultilanguageReferrable, Referrable , ServiceNeeds			
Attribute	Datatype	Mul.	Kind	Note
activateAtStart	Boolean	1	attr	True/false: supervision activation status of SupervisedEntity shall be enabled/disabled at start.
enableDeactivation	Boolean	1	attr	True: software-component shall be allowed to deactivate supervision of this SupervisedEntity false: software-component shall be not allowed to deactivate supervision of this SupervisedEntity
expectedAliveCycle	TimeValue	1	attr	Expected cycle time of alive trigger of this SupervisedEntity (in seconds).
maxAliveCycle	TimeValue	1	attr	Maximum cycle time of alive trigger of this SupervisedEntity (in seconds).
minAliveCycle	TimeValue	1	attr	Minimum cycle time of alive trigger of this SupervisedEntity (in seconds).
toleratedFailedCycles	PositiveInteger	1	attr	Number of consecutive failed alive cycles for this SupervisedEntity which shall be tolerated until the supervision status of the SupervisedEntity is set to WdGM_ALIVE_EXPIRED (see SWS WdGM for more details). Note that this value has to be recalculated with respect to the WdGM's own cycle time for ECU configuration.

Table 7.60: SupervisedEntityNeeds

Scenario: an [AtomicSwComponentType](#) contains a *Supervised Entity*. In this case it is required that the *Supervised Entity* indicates to the *Watchdog Manager* that a Checkpoint within the *Supervised Entity* has been reached. Further on the *Local Supervision*

Status of a single *Supervised Entity* may be signaled to the software component. In this case the following setup applies:

[TPS_SWCT_02018] Setup for *AtomicSwComponentType* which contains a *Supervised Entity* [

RoleBasedPortAssignment valid roles:

- WdgM_AliveSupervision [1]
- WdgM_IndividualMode [0..1]

RoleBasedDataAssignment

N/A

RepresentedPortGroups

N/A

]()

For more information please refer to [SWS_WdgM_00333], and [SWS_WdgM_00335].

Please note that an *SwcInternalBehavior* may provide several *SupervisedEntityNeeds* elements where each defines the requirements in relation to one supervised entity.

7.11.3.2.2 Watchdog Service use Case: *Global Supervision Status* notification

Scenario: an *AtomicSwComponentType* requires to receive the *Global Supervision Status* that is combined from all individual *Supervised Entities*. In this case the following setup applies:

[TPS_SWCT_02019] Setup for *AtomicSwComponentType* which requires *Global Supervision Status* notification [

RoleBasedPortAssignment valid roles:

- WdgM_GlobalMode [1]

RoleBasedDataAssignment

N/A

RepresentedPortGroups

N/A

]()

For more information please refer to [SWS_WdgM_00336].

7.11.3.3 COM Manager Service Needs

The meta-class `ComMgrUserNeeds` is used to define requirements to configure the ComM Service. An `SwcInternalBehavior` may provide several `ComMgrUserNeeds` elements where each defines the requirements from one “user” of the ComM Service. Especially, it defines which `PortGroup` is associated with this “user”.

Class	ComMgrUserNeeds			
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
Note	Specifies the abstract needs on the configuration of the Communication Manager for one "user".			
Base	ARObject, Identifiable , MultilanguageReferrable , Referrable , ServiceNeeds			
Attribute	Datatype	Mul.	Kind	Note
maxComm Mode	MaxCommModeEnum	1	attr	Maximum communication mode requested by this ComM user.

Table 7.61: ComMgrUserNeeds

7.11.3.3.1 ComM Use Case: read current ComM Mode

Scenario: a `AtomicSwComponentType` reads the current ComM mode.

In this case the following rules apply:

[TPS_SWCT_01019] `AtomicSwComponentType` reads the current ComM mode [

RoleBasedPortAssignment valid roles:

- `ComM_CurrentMode` [1]

RoleBasedDataAssignment

N/A

RepresentedPortGroup

N/A

]()

For more information please refer to [SWS_ComM_00847].

7.11.3.3.2 ComM Use Case: request ComM Mode

Scenario: a `AtomicSwComponentType` requests a ComM mode. It may also check later whether the requested ComM mode has become effective.

In this case the following rules apply:

[TPS_SWCT_01020] `AtomicSwComponentType` requests a ComM mode. It may also check later whether the requested ComM mode has become effective [

RoleBasedPortAssignment valid roles:

- ComM_CurrentMode [1]
- ComM_UserRequest [1]

RoleBasedDataAssignment

N/A

RepresentedPortGroup

Reference to the applicable [PortGroup](#) [0..1]

]()

For more information please refer to [SWS_ComM_00848].

7.11.3.3.3 ComM Use Case: Software-Component acts as a Mode Manager that influences the ECU State

Scenario: a [AtomicSwComponentType](#) acts as a mode manager that influences the ECU state.

In this case the following rules apply:

[TPS_SWCT_01021] [AtomicSwComponentType](#) acts as a mode manager that influences the ECU state [

RoleBasedPortAssignment valid roles:

- ComM_CurrentMode [0..1]
- ComM_UserRequest [0..1]
- ComM_ECUModeLimitation [1]

RoleBasedDataAssignment

N/A

RepresentedPortGroup

N/A

]()

For more information please refer to [SWS_ComM_00741].

7.11.3.4 ECU State Manager Service Needs

The meta-class [EcuStateMgrUserNeeds](#) is used to define the requirements to configure the ECU State Manager Service. There are actually two variants of AUTOSAR ECU management: flexible and fixed. An [SwcInternalBehavior](#) may provide several [EcuStateMgrUserNeeds](#) elements where each defines the requirements from

one “user” of the EcuM Service (for the terms related to the AUTOSAR ECU State Manager see [33]).

Class	EcuStateMgrUserNeeds			
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
Note	Specifies the abstract needs on the configuration of the ECU State Manager for one "user". This class currently contains no attributes. Its name can be regarded as a symbol identifying the user from the viewpoint of the component or module which owns this class.			
Base	ARObject, Identifiable , MultilanguageReferrable, Referrable , ServiceNeeds			
Attribute	Datatype	Mul.	Kind	Note
–	–	–	–	–

Table 7.62: EcuStateMgrUserNeeds

7.11.3.4.1 EcuM Fixed Use Case: read current ECU Mode

Scenario: a [AtomicSwComponentType](#) reads the current ECU mode.

In this case the following rules apply:

[TPS_SWCT_01012] [AtomicSwComponentType](#) reads the current ECU mode (fixed variant) [

RoleBasedPortAssignment valid roles:

- EcuM_CurrentMode [1]

RoleBasedDataAssignment

N/A

RepresentedPortGroup

N/A

]()

For more information please refer to [SWS_EcuM_02762] and [SWS_EcuM_02749].

7.11.3.4.2 EcuM Fixed Use Case: request a certain ECU state

Scenario: a [AtomicSwComponentType](#) needs to keep the ECU alive or needs to execute operations before the ECU is shut down. For this purpose the [AtomicSwComponentType](#) may request either the state RUN or POST_RUN.

In this case the following rules apply:

[TPS_SWCT_01013] [AtomicSwComponentType](#) shall keep the ECU alive (fixed variant) [

[AtomicSwComponentType](#) needs to keep the ECU alive or needs to execute operations before the ECU is shut down.

RoleBasedPortAssignment valid roles:

- [EcuM_StateRequest](#) [1]

RoleBasedDataAssignment

N/A

RepresentedPortGroup

N/A

]()

For more information please refer to [SWS_EcuM_02762].

7.11.3.4.3 EcuM Fixed Use Case: select Shutdown Target

Scenario: a [AtomicSwComponentType](#) wants to select a shutdown target. This corresponds to the “select shutdown target” use case of the flex EcuM.

In this case the following rules apply:

[TPS_SWCT_01014] [AtomicSwComponentType](#) wants to select a shutdown target (fixed variant) [

RoleBasedPortAssignment valid roles:

- [EcuM_ShutdownTarget](#) [1]

RoleBasedDataAssignment

N/A

RepresentedPortGroup

N/A

]()

7.11.3.4.4 EcuM Fixed Use Case: select Boot Target

Scenario: a [AtomicSwComponentType](#) wants to select a boot target.

In this case the following rules apply:

[TPS_SWCT_01015] [AtomicSwComponentType](#) wants to select a boot target (fixed variant) [

RoleBasedPortAssignment valid roles:

- [EcuM_BootTarget](#) [1]

RoleBasedDataAssignment

N/A

RepresentedPortGroup

N/A

]()

7.11.3.4.5 EcuM Flex Use Case: select Shutdown Target

Scenario: a [AtomicSwComponentType](#) wants to select a shutdown target. This corresponds to the “select shutdown target” use case of the fix EcuM.

In this case the following rules apply:

[TPS_SWCT_01016] [AtomicSwComponentType](#) wants to select a shutdown target (flexible variant) [

RoleBasedPortAssignment valid roles:

- EcuM_ShutdownTarget [1]

RoleBasedDataAssignment

N/A

RepresentedPortGroup

N/A

]()

7.11.3.4.6 EcuM Flex Use Case: select Boot Target

Scenario: a [AtomicSwComponentType](#) wants to select a boot target.

In this case the following rules apply:

[TPS_SWCT_01017] [AtomicSwComponentType](#) wants to select a boot target (flexible variant) [

RoleBasedPortAssignment valid roles:

- EcuM_BootTarget [1]

RoleBasedDataAssignment

N/A

RepresentedPortGroup

N/A

]()

7.11.3.4.7 EcuM Flex Use Case: use Alarm Clock

Scenario: a `AtomicSwComponentType` wants to use an alarm clock.

In this case the following rules apply:

[TPS_SWCT_01018] `AtomicSwComponentType` wants to use an alarm clock (flexible variant) [

RoleBasedPortAssignment valid roles:

- `EcuM_AlarmClock` [1]

RoleBasedDataAssignment

N/A

RepresentedPortGroup

N/A

]()

7.11.3.5 BswM

All use cases for interaction of an application software-component with the `BswM` require the aggregation in the role `serviceNeeds` of `BswMgrNeeds`, a subclass of `ServiceNeeds`, at `SwcServiceDependency`.

Class	BswMgrNeeds			
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
Note	Specifies the abstract needs on the configuration of the Basic Software Manager for one "user".			
Base	ARObject, Identifiable , MultilanguageReferrable, Referrable , ServiceNeeds			
Attribute	Datatype	Mul.	Kind	Note
—	—	—	—	—

Table 7.63: BswMgrNeeds

7.11.3.5.1 Partial Networking

One specific use case for the existence of a `SwcServiceDependency` with respect to the interaction with the `BswM` is the support for partial networking, in particular the association of a `PortGroup` and the associated `PortPrototypes` that act as *VFC control ports* and *VFC status ports*. For more details please refer to section 4.8.

In this case the following rules apply:

[TPS_SWCT_01126] Access to partial networking via BswM [

RoleBasedPortAssignment valid roles:

- control [0 .. 1]
- status [0 .. 1]

RoleBasedDataAssignment

N/A

RepresentedPortGroup

Reference to the applicable [PortGroup](#) associated with the particular partial network.

]([RS_SWCT_03201](#))

The multiplicities of the [RoleBasedPortAssignments](#) for this case have been defined under the assumption that a given software-component may or may not have a *VFC control port*. Also, it may have a *VFC status port*. Technically, there could be several *VFC status ports* per software-component but most likely there is only one *VFC status port*.

7.11.3.5.2 Mode Manager

A software-component that acts as a mode manager exposes a [PPortPrototype](#) typed by a [ModeSwitchInterface](#). By this means the mode manager communicates changes of the particular mode to the connected mode users.

On the side of the `BswM`, an [RPortPrototype](#) typed by an [ModeSwitchInterface](#) used to receive notifications of mode switches will have to be established (for more details, please refer to [SWS_BswM_00196] and [SWS_BswM_00200]).

In this case the following rules apply:

[TPS_SWCT_01552] Software-component acts as a mode manager [

RoleBasedPortAssignment valid roles:

- AppModeInterface [1]

RoleBasedDataAssignment

N/A

RepresentedPortGroup

none.

]([RS_SWCT_03110](#), [RS_SWCT_03200](#), [RS_SWCT_03203](#))

A slight variation of this use case exists if the Application Mode Manager serves mode users that consist of both the `BswM` and other [ApplicationSwComponentTypes](#).

[TPS_SWCT_01572] Application Mode Manager interacts with both `BswM` and other [ApplicationSwComponentTypes](#) [If an Application Mode Manager interacts with both `BswM` and other [ApplicationSwComponentTypes](#) the following requirements on the modeling of this scenario shall be taken into account:

Mode Request For the configuration of mode requests two separate `AbstractRequiredPortPrototypes` shall exist:

- One `AbstractRequiredPortPrototype` shall be typed by a `SenderReceiverInterface` with attribute `isService` set to `true`. This `AbstractRequiredPortPrototype` shall be connected to the `SwComponentPrototype` typed by a `ServiceSwComponentType` representing the BswM.
- One `AbstractRequiredPortPrototype` shall be typed by a `SenderReceiverInterface` with attribute `isService` set to `false`. This `AbstractRequiredPortPrototype` shall be connected to `SwComponentPrototypes` typed by `ApplicationSwComponentTypes` that request model changes.

Mode Switch Notification An Application Mode Manager that sends mode switch notifications to both BswM and other `ApplicationSwComponentTypes` shall expose a single `AbstractProvidedPortPrototype` for sending the mode switch notification to both the BswM and `ApplicationSwComponentTypes`.

The value of the attribute `ModeSwitchInterface.isService` shall be set to `false`).

|(RS_SWCT_03200, RS_SWCT_03202)

Rationale for [TPS_SWCT_01572]: technically, the existence of two separate `AbstractProvidedPortPrototype` for sending the mode switch notification to both the BswM and `ApplicationSwComponentTypes` would end up in two separate mode machines in the RTE and it would be a tough challenge to keep both mode machines perfectly synchronized.

Therefore, the exception regarding the usage of the attribute `isService` is justified to mitigate this effect.

On the mode request side, however, the situation is entirely different because the mode requests need arbitration by the Application Mode Manager anyway. This is completely in the scope of the implementation of the Application Mode Manager and AUTOSAR has no stakes in further standardizing this aspect.

Therefore, there is no motivation for a further exception with respect to the value of `isService`.

7.11.3.5.3 Mode User

A software-component that acts as a mode user exposes an `RPortPrototype` typed by a `ModeSwitchInterface`. By this means the software-component can be notified by mode switches executed at the mode manager (in this case the BswM).

On the side of the `BswM`, an `PPortPrototype` typed by an `ModeSwitchInterface` used to send out notifications of mode switches will have to be established (for more details, please refer to [SWS_BswM_00196] and [SWS_BswM_00202]).

In this case the following rules apply:

[TPS_SWCT_01553] Software-component acts as a mode user [

RoleBasedPortAssignment valid roles:

- `AppModeInterface` [1]

RoleBasedDataAssignment

N/A

RepresentedPortGroup

none.

]([RS_SWCT_03110](#), [RS_SWCT_03200](#), [RS_SWCT_03203](#))

7.11.3.5.4 Mode Requester

A software-component that acts as a mode requester exposes an `PPortPrototype` typed by a `SenderReceiverInterface`. By this means the software-component can send mode requests towards the mode manager (in this case the `BswM`).

On the side of the `BswM`, an `RPortPrototype` typed by an `SenderReceiverInterface` used to requests for mode switches will have to be established (for more details, please refer to [SWS_BswM_00195] and [SWS_BswM_00201]).

In this case the following rules apply:

[TPS_SWCT_01554] Software-component acts as a mode requester [

RoleBasedPortAssignment valid roles:

- `AppModeRequestInterface` [1]

RoleBasedDataAssignment

N/A

RepresentedPortGroup

none.

]([RS_SWCT_03110](#), [RS_SWCT_03200](#), [RS_SWCT_03202](#))

7.11.3.6 Crypto Service Dependencies

The meta-class `CryptoServiceNeeds` is used to define the requirements to configure the `CryptoServiceManager`.

An [SwcInternalBehavior](#) may provide several [CryptoServiceNeeds](#) elements where each relates to one ConfigID (see [34]) for details). In this context it is of special importance to note which [PortPrototypes](#) belong to this ConfigID in order to be able to properly generate the callbacks.

Class	CryptoServiceNeeds			
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
Note	Specifies the needs on the configuration of the CryptoServiceManager for one ConfigID (see Specification AUTOSAR_SWS_CSM.doc). An instance of this class is used to find out which ports of an SWC belong to this ConfigID.			
Base	ARObject, Identifiable , MultilanguageReferrable, Referrable , ServiceNeeds			
Attribute	Datatype	Mul.	Kind	Note
maximumKeyLength	PositiveInteger	0..1	attr	The maximum length of a cryptographic key, that is used by the SWC or module for this configuration. Unit: bit.

Table 7.64: CryptoServiceNeeds

Please note that for all described use cases of the Crypto Service following rule applies:

For every used [ClientServerInterface](#) it is necessary to create a [RoleBasedPortAssignment](#). Thereby the value of the attribute `role` of the [RoleBasedPortAssignment](#) has to be set to the name of the used standardized [ClientServerInterface](#). The possible role attribute values and the multiplicity of the related [PortPrototypes](#) are listed at the use case descriptions in the paragraph **RoleBasedPortAssignment**.

7.11.3.6.1 Crypto Service Service Use Case: Hash calculation

Scenario: a [AtomicSwComponentType](#) uses the hash calculation of the Crypto Service. In this case the following setup apply:

[TPS_SWCT_02020] [AtomicSwComponentType](#) uses the hash calculation of the Crypto Service [

RoleBasedPortAssignment valid roles:

- CsmHash [1]
- CsmCallback [1]

RoleBasedDataAssignment

N/A

RepresentedPortGroups

N/A

]()

For more information please refer to [SWS_Csm_00775] and [SWS_Csm_00801].

7.11.3.6.2 Crypto Service Service Use Case: MAC calculation

Scenario: a `AtomicSwComponentType` uses the message authentication code (MAC) calculation of the Crypto Service. In this case the following setup apply:

[TPS_SWCT_02021] `AtomicSwComponentType` uses the message authentication code (MAC) calculation of the Crypto Service [

RoleBasedPortAssignment valid roles:

- `CsmMacGenerate` [1]
- `CsmCallback` [1]

RoleBasedDataAssignment

N/A

RepresentedPortGroups

N/A

]()

For more information please refer to [SWS_Csm_00776] and [SWS_Csm_00801].

7.11.3.6.3 Crypto Service Service Use Case: MAC verification

Scenario: a `AtomicSwComponentType` uses the message authentication code (MAC) verification of the Crypto Service. In this case the following setup apply:

[TPS_SWCT_02022] `AtomicSwComponentType` uses the message authentication code (MAC) verification of the Crypto Service [

RoleBasedPortAssignment valid roles:

- `CsmMacVerify` [1]
- `CsmCallback` [1]

RoleBasedDataAssignment

N/A

RepresentedPortGroups

N/A

]()

For more information please refer to [SWS_Csm_00777] and [SWS_Csm_00801].

7.11.3.6.4 Crypto Service Service Use Case: seeding of random generator

Scenario: a `AtomicSwComponentType` uses the generation of random numbers of the Crypto Service. In this case the following setup apply:

[TPS_SWCT_02023] `AtomicSwComponentType` uses the generation of random seed of the Crypto Service [

RoleBasedPortAssignment valid roles:

- `CsmRandomSeed` [1]
- `CsmCallback` [1]

RoleBasedDataAssignment

N/A

RepresentedPortGroups

N/A

]()

For more information please refer to [SWS_Csm_00778] and [SWS_Csm_00801].

7.11.3.6.5 Crypto Service Service Use Case: generation of random numbers

Scenario: a `AtomicSwComponentType` uses the generation of random numbers of the Crypto Service. In this case the following setup apply:

[TPS_SWCT_02024] `AtomicSwComponentType` uses the generation of random numbers of the Crypto Service [

RoleBasedPortAssignment valid roles:

- `CsmRandomGenerate` [1]
- `CsmCallback` [1]

RoleBasedDataAssignment

N/A

RepresentedPortGroups

N/A

]()

For more information please refer to [SWS_Csm_00779] and [SWS_Csm_00801].

7.11.3.6.6 Crypto Service Service Use Case: symmetrical block encryption

Scenario: a [AtomicSwComponentType](#) uses the symmetrical block encryption of the Crypto Service. In this case the following setup apply:

[TPS_SWCT_02025] [AtomicSwComponentType](#) uses the symmetrical block encryption of the Crypto Service [

RoleBasedPortAssignment valid roles:

- CsmSymBlockEncrypt [1]
- CsmCallback [1]

RoleBasedDataAssignment

N/A

RepresentedPortGroups

N/A

]()

For more information please refer to [SWS_Csm_00780] and [SWS_Csm_00801].

7.11.3.6.7 Crypto Service Service Use Case: symmetrical block decryption

Scenario: a [AtomicSwComponentType](#) uses the symmetrical block decryption of the Crypto Service. In this case the following setup apply:

[TPS_SWCT_02026] [AtomicSwComponentType](#) uses the symmetrical block decryption of the Crypto Service [

RoleBasedPortAssignment valid roles:

- CsmSymBlockDecrypt [1]
- CsmCallback [1]

RoleBasedDataAssignment

N/A

RepresentedPortGroups

N/A

]()

For more information please refer to [SWS_Csm_00781] and [SWS_Csm_00801].

7.11.3.6.8 Crypto Service Service Use Case: symmetrical encryption

Scenario: a [AtomicSwComponentType](#) uses the symmetrical encryption of the Crypto Service. In this case the following setup apply:

[TPS_SWCT_02027] [AtomicSwComponentType](#) uses the symmetrical encryption of the Crypto Service [

RoleBasedPortAssignment valid roles:

- CsmSymEncrypt [1]
- CsmCallback [1]

RoleBasedDataAssignment

N/A

RepresentedPortGroups

N/A

]()

For more information please refer to [SWS_Csm_00782] and [SWS_Csm_00801].

7.11.3.6.9 Crypto Service Service Use Case: symmetrical decryption

Scenario: a [AtomicSwComponentType](#) uses the symmetrical decryption of the Crypto Service. In this case the following setup apply:

[TPS_SWCT_02028] [AtomicSwComponentType](#) uses the symmetrical decryption of the Crypto Service [

RoleBasedPortAssignment valid roles:

- CsmSymDecrypt [1]
- CsmCallback [1]

RoleBasedDataAssignment

N/A

RepresentedPortGroups

N/A

]()

For more information please refer to [SWS_Csm_00783] and [SWS_Csm_00801].

7.11.3.6.10 Crypto Service Service Use Case: asymmetrical encryption

Scenario: a [AtomicSwComponentType](#) uses the asymmetrical encryption of the Crypto Service. In this case the following setup apply:

[TPS_SWCT_02029] [AtomicSwComponentType](#) uses the asymmetrical encryption of the Crypto Service [

RoleBasedPortAssignment valid roles:

- CsmAsymEncrypt [1]
- CsmCallback [1]

RoleBasedDataAssignment

N/A

RepresentedPortGroups

N/A

]()

For more information please refer to [SWS_Csm_00784] and [SWS_Csm_00801].

7.11.3.6.11 Crypto Service Service Use Case: asymmetrical decryption

Scenario: a [AtomicSwComponentType](#) uses the asymmetrical decryption of the Crypto Service. In this case the following setup apply:

[TPS_SWCT_02030] [AtomicSwComponentType](#) uses the asymmetrical decryption of the Crypto Service [

RoleBasedPortAssignment valid roles:

- CsmAsymDecrypt [1]
- CsmCallback [1]

RoleBasedDataAssignment

N/A

RepresentedPortGroups

N/A

]()

For more information please refer to [SWS_Csm_00785] and [SWS_Csm_00801].

7.11.3.6.12 Crypto Service Service Use Case: signature generation

Scenario: a [AtomicSwComponentType](#) uses the signature generation of the Crypto Service. In this case the following setup apply:

[TPS_SWCT_02031] [AtomicSwComponentType](#) uses the signature generation of the Crypto Service [

RoleBasedPortAssignment valid roles:

- CsmSignatureGenerate [1]
- CsmCallback [1]

RoleBasedDataAssignment

N/A

RepresentedPortGroups

N/A

]()

For more information please refer to [SWS_Csm_00786] and [SWS_Csm_00801].

7.11.3.6.13 Crypto Service Service Use Case: signature verification

Scenario: a [AtomicSwComponentType](#) uses the signature verification of the Crypto Service. In this case the following setup apply:

[TPS_SWCT_02032] [AtomicSwComponentType](#) uses the signature verification of the Crypto Service [

RoleBasedPortAssignment valid roles:

- CsmSignatureVerify [1]
- CsmCallback [1]

RoleBasedDataAssignment

N/A

RepresentedPortGroups

N/A

]()

For more information please refer to [SWS_Csm_00787] and [SWS_Csm_00801].

7.11.3.6.14 Crypto Service Service Use Case: checksum calculation

Scenario: a [AtomicSwComponentType](#) uses the checksum calculation of the Crypto Service. In this case the following setup apply:

[TPS_SWCT_02033] [AtomicSwComponentType](#) uses the checksum calculation of the Crypto Service [

RoleBasedPortAssignment valid roles:

- CsmChecksum [1]
- CsmCallback [1]

RoleBasedDataAssignment

N/A

RepresentedPortGroups

N/A

]()

For more information please refer to [SWS_Csm_00788] and [SWS_Csm_00801].

7.11.3.6.15 Crypto Service Service Use Case: key derivation

Scenario: a [AtomicSwComponentType](#) uses the key derivation of the Crypto Service. In this case the following setup apply:

[TPS_SWCT_02034] [AtomicSwComponentType](#) uses the key derivation of the Crypto Service [

RoleBasedPortAssignment valid roles:

- CsmKeyDerive [1]
- CsmCallback [1]

RoleBasedDataAssignment

N/A

RepresentedPortGroups

N/A

]()

For more information please refer to [SWS_Csm_00789] and [SWS_Csm_00801].

7.11.3.6.16 Crypto Service Service Use Case: symmetric key derivation

Scenario: a [AtomicSwComponentType](#) uses the symmetric key derivation of the Crypto Service. In this case the following setup apply:

[TPS_SWCT_02035] [AtomicSwComponentType](#) uses the symmetric key derivation of the Crypto Service [

RoleBasedPortAssignment valid roles:

- CsmKeyDeriveSymKey [1]
- CsmCallback [1]

RoleBasedDataAssignment

N/A

RepresentedPortGroups

N/A

]()

For more information please refer to [SWS_Csm_00790] and [SWS_Csm_00801].

7.11.3.6.17 Crypto Service Service Use Case: key exchange protocol, public value calculation

Scenario: a [AtomicSwComponentType](#) uses the key exchange interface for public value calculation of the Crypto Service. In this case the following setup apply:

[TPS_SWCT_02036] [AtomicSwComponentType](#) uses the key exchange interface for public value calculation of the Crypto Service [

RoleBasedPortAssignment valid roles:

- CsmKeyExchangeCalcPubVal [1]
- CsmCallback [1]

RoleBasedDataAssignment

N/A

RepresentedPortGroups

N/A

]()

For more information please refer to [SWS_Csm_00791] and [SWS_Csm_00801].

7.11.3.6.18 Crypto Service Service Use Case: key exchange protocol, secret value calculation

Scenario: a [AtomicSwComponentType](#) uses the key exchange interface for secret value calculation of the Crypto Service. In this case the following setup apply:

[TPS_SWCT_02037] [AtomicSwComponentType](#) uses the key exchange interface for secret value calculation of the Crypto Service [

RoleBasedPortAssignment valid roles:

- CsmKeyExchangeCalcSecret [1]
- CsmCallback [1]

RoleBasedDataAssignment

N/A

RepresentedPortGroups

N/A

]()

For more information please refer to [SWS_Csm_00792] and [SWS_Csm_00801].

7.11.3.6.19 Crypto Service Service Use Case: key exchange protocol, calculate symmetric key

Scenario: a [AtomicSwComponentType](#) uses the key exchange interface to calculate symmetric key with the Crypto Service. In this case the following setup apply:

[TPS_SWCT_02038] [AtomicSwComponentType](#) uses the key exchange interface to calculate symmetric key with the Crypto Service [

RoleBasedPortAssignment valid roles:

- CsmKeyExchangeCalcSymKey [1]
- CsmCallback [1]

RoleBasedDataAssignment

N/A

RepresentedPortGroups

N/A

]()

For more information please refer to [SWS_Csm_00793] and [SWS_Csm_00801].

7.11.3.6.20 Crypto Service Service Use Case: symmetrical key extraction

Scenario: a [AtomicSwComponentType](#) uses the symmetrical key extraction of the Crypto Service. In this case the following setup apply:

[TPS_SWCT_02039] [AtomicSwComponentType](#) uses the symmetrical key extraction of the Crypto Service [

RoleBasedPortAssignment valid roles:

- CsmSymKeyExtract [1]
- CsmCallback [1]

RoleBasedDataAssignment

N/A

RepresentedPortGroups

N/A

]()

For more information please refer to [SWS_Csm_00794] and [SWS_Csm_00801].

7.11.3.6.21 Crypto Service Service Use Case: symmetrical key wrapping with symmetrical wrapping key

Scenario: a [AtomicSwComponentType](#) uses the symmetrical key wrapping of the Crypto Service to export a symmetrical key structure with a symmetric key. In this case the following setup apply:

[TPS_SWCT_02040] [AtomicSwComponentType](#) uses the symmetrical key wrapping of the Crypto Service to export a symmetrical key structure with a symmetric key [

RoleBasedPortAssignment valid roles:

- CsmSymKeyWrapSym [1]
- CsmCallback [1]

RoleBasedDataAssignment

N/A

RepresentedPortGroups

N/A

]()

For more information please refer to [SWS_Csm_00795] and [SWS_Csm_00801].

7.11.3.6.22 Crypto Service Service Use Case: symmetrical key wrapping with asymmetrical wrapping key

Scenario: a [AtomicSwComponentType](#) uses the asymmetrical key wrapping of the Crypto Service to export a symmetrical key structure with a asymmetric key. In this case the following setup apply:

[TPS_SWCT_02041] [AtomicSwComponentType](#) uses the asymmetrical key wrapping of the Crypto Service to export a symmetrical key structure with a asymmetric key [

RoleBasedPortAssignment valid roles:

- CsmSymKeyWrapAsym [1]
- CsmCallback [1]

RoleBasedDataAssignment

N/A

RepresentedPortGroups

N/A

]()

For more information please refer to [SWS_Csm_00796] and [SWS_Csm_00801].

7.11.3.6.23 Crypto Service Service Use Case: asymmetrical public key extraction

Scenario: a [AtomicSwComponentType](#) uses the asymmetrical public key extraction of the Crypto Service. In this case the following setup apply:

[TPS_SWCT_02042] [AtomicSwComponentType](#) uses the asymmetrical public key extraction of the Crypto Service [

RoleBasedPortAssignment valid roles:

- CsmAsymPublicKeyExtract [1]
- CsmCallback [1]

RoleBasedDataAssignment

N/A

RepresentedPortGroups

N/A

]()

For more information please refer to [SWS_Csm_00797] and [SWS_Csm_00801].

7.11.3.6.24 Crypto Service Service Use Case: asymmetrical private key extraction

Scenario: a [AtomicSwComponentType](#) uses the asymmetrical private key extraction of the Crypto Service. In this case the following setup apply:

[TPS_SWCT_02043] [AtomicSwComponentType](#) uses the asymmetrical private key extraction of the Crypto Service [

RoleBasedPortAssignment valid roles:

- CsmAsymPrivateKeyExtract [1]
- CsmCallback [1]

RoleBasedDataAssignment
N/A

RepresentedPortGroups
N/A

]()

For more information please refer to [SWS_Csm_00798] and [SWS_Csm_00801].

7.11.3.6.25 Crypto Service Service Use Case: asymmetrical key wrapping with symmetrical wrapping key

Scenario: a [AtomicSwComponentType](#) uses the asymmetrical key wrapping of the Crypto Service to export a (asymmetric) private key structure with a symmetrical wrapping key. In this case the following setup apply:

[TPS_SWCT_02044] [AtomicSwComponentType](#) uses the asymmetrical key wrapping of the Crypto Service to export a (asymmetric) private key structure with a symmetrical wrapping key [

RoleBasedPortAssignment valid roles:

- CsmAsymPrivateKeyWrapSym [1]
- CsmCallback [1]

RoleBasedDataAssignment
N/A

RepresentedPortGroups
N/A

]()

For more information please refer to [SWS_Csm_00799] and [SWS_Csm_00801].

7.11.3.6.26 Crypto Service Service Use Case: asymmetrical key wrapping with asymmetrical wrapping key

Scenario: a [AtomicSwComponentType](#) uses the asymmetrical key wrapping of the Crypto Service to export a (asymmetric) private key structure with a asymmetrical wrapping key. In this case the following setup apply:

[TPS_SWCT_02045] [AtomicSwComponentType](#) uses the asymmetrical key wrapping of the Crypto Service to export a (asymmetric) private key structure with a asymmetrical wrapping key [

RoleBasedPortAssignment valid roles:

- [CsmAsymPrivateKeyWrapAsym](#) [1]
- [CsmCallback](#) [1]

RoleBasedDataAssignment

N/A

RepresentedPortGroups

N/A

]()

For more information please refer to [SWS_Csm_00800] and [SWS_Csm_00801].

7.11.3.7 Diagnostic Service Dependency

This chapter describes the usage of the specific diagnostic meta-classes derived from [ServiceNeeds](#) within an atomic software-component. An overview of common diagnostic service needs has already been introduced in figure 7.36 and can be divided into four main parts: Function Inhibition Needs 7.11.3.7.1, Diagnostic Event Needs 7.11.3.7.2, Diagnostic Communication Needs 7.11.3.7.3, and needs to fulfill the OBD related requirements 7.11.3.7.4.

Please note that for the described use cases of the Diagnostic Services the following rule applies:

[TPS_SWCT_01129] Express diagnostic capabilities [For every used [ClientServerInterface](#) it is necessary to create a [RoleBasedPortAssignment](#). Thereby the value of the attribute [role](#) of the [RoleBasedPortAssignment](#) has to be set to the name of the used standardized [ClientServerInterface](#).

The possible role attribute values and the multiplicity of the related [PortPrototypes](#) are listed at the use case descriptions in the paragraph **RoleBasedPortAssignment**.
]([RS_SWCT_03190](#))

7.11.3.7.1 Function Inhibition Needs

The meta-class `FunctionInhibitionNeeds` is used to define requirements in order to configure the Diagnostic Event Manager Service.

An `SwcInternalBehavior` may provide several `FunctionInhibitionNeeds` elements, each defines the requirements related to one function inhibition ID (for the terms related to the AUTOSAR Function Inhibition Manager, see [35]).

Class	FunctionInhibitionNeeds			
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
Note	Specifies the abstract needs on the configuration of the Function Inhibition Manager for one Function Identifier (FID). This class currently contains no attributes. Its name can be regarded as a symbol identifying the FID from the viewpoint of the component or module which owns this class.			
Base	ARObject, <code>Identifiable</code> , <code>MultilanguageReferrable</code> , <code>Referrable</code> , <code>ServiceNeeds</code>			
Attribute	Datatype	Mul.	Kind	Note
–	–	–	–	–

Table 7.65: FunctionInhibitionNeeds

7.11.3.7.1.1 Function Inhibition Manager Service use Case: read function permission

[TPS_SWCT_02505] Setup for Function Inhibition Manager Service use Case: read function permission | Scenario: a `AtomicSwComponentType` read the function permission from FiM in order to enable or disable a functionality. In this case the following setup apply:

ServiceNeeds kind `FunctionInhibitionNeeds`

RoleBasedPortAssignment valid roles:

- `FunctionInhibition` [1]

RoleBasedDataAssignment

N/A

RepresentedPortGroups

N/A

]()

For more information please refer to [SWS_Fim_00090].

7.11.3.7.2 Diagnostic Event Needs

The meta-classes `DiagnosticEventManagerNeeds` is used to define requirements in order to configure the Diagnostic Event Manager Service.

An `SwcInternalBehavior` may provide several `DiagnosticEventManagerNeeds` elements that define the mappings for the general diagnostic event manager behavior (for the terms related to the AUTOSAR Diagnostic Event Manager see [36]).

Class	DiagnosticEventManagerNeeds			
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
Note	Specifies the general needs on the configuration of the Diagnostic Event Manager (DEM) which are not related to a particular item.			
Base	ARObject, DiagnosticCapabilityElement , Identifiable , Multilanguage Referrable, Referrable , ServiceNeeds			
Attribute	Datatype	Mul.	Kind	Note
—	—	—	—	—

Table 7.66: DiagnosticEventManagerNeeds

The meta-class `DiagnosticCapabilityElement` is used to provide generic information about diagnostic capabilities. Further on, the usage of `DiagnosticCapabilityElement` indicates that all `ServiceNeeds` which inherit from `DiagnosticCapabilityElement` express the following intentions:

- Need to interact with AUTOSAR Service Dem or Dcm.
- Provide services for the on-board diagnostics.

Class	DiagnosticCapabilityElement (abstract)			
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
Note	This class identifies the capability to provide generic information about diagnostic capabilities			
Base	ARObject, Identifiable , MultilanguageReferrable, Referrable , ServiceNeeds			
Attribute	Datatype	Mul.	Kind	Note
audience	DiagnosticAudienceEnum	*	attr	This specifies the intended audience for the diagnostic object. Note that this is not only for the documentation but also subsequent audience specific implementation.
diagRequirement	DiagRequirementIdString	0..1	attr	This denotes the requirement identifier to which the object can be linked to. Note that with the implementation of a generic tracing concept in AUTOSAR this attribute might become obsolete.
securityAccessLevel	PositiveInteger	0..1	attr	This attribute denotes the level of security which is touched by the diagnostic object. The higher the level the more relevance for the security exists. This level shall be mapped to the security level in the ECU.

Table 7.67: DiagnosticCapabilityElement

Enumeration	DiagnosticAudienceEnum
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds
Note	The possible values of the intended audience for a diagnostic object.
Literal	Description
afterSales	The object is relevant for the OEM after-sales organization.
aftermarket	The object is for free aftermarket service organizations.
development	The object is relevant for engineering only.
manufacturing	The object is relevant for manufacturing.
supplier	The object is relevant for the ECU-supplier aftermarket organization.

Table 7.68: DiagnosticAudienceEnum

The meta-classes [DiagnosticEventNeeds](#) is used to define requirements to configure the Diagnostic Event Manager Service. An [SwcInternalBehavior](#) may provide several [DiagnosticEventNeeds](#) elements where each defines all the requirements related to one diagnostic event (for the terms related to the AUTOSAR Diagnostic Event Manager see [36]).

In addition, [ObdPidServiceNeeds](#) and [ObdRatioServiceNeeds](#) are required in order to specify the needs for OBD diagnostic service calls.

[TPS_SWCT_01591] Existence of attribute [DiagnosticEventNeeds.reportBehavior](#) [The attribute [DiagnosticEventNeeds.reportBehavior](#) shall be ignored if it is specified in the context of a [SwcServiceDependency](#).]([RS_SWCT_03190](#))

The rationale for the existence of [\[TPS_SWCT_01591\]](#) is that a software-component can never report errors to the Dem **before** the Dem is fully initialized.

Class	DiagnosticEventManagerNeeds			
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
Note	Specifies the general needs on the configuration of the Diagnostic Event Manager (DEM) which are not related to a particular item.			
Base	ARObject, DiagnosticCapabilityElement , Identifiable ,Multilanguage Referrable, Referrable , ServiceNeeds			
Attribute	Datatype	Mul.	Kind	Note
—	—	—	—	—

Table 7.69: DiagnosticEventManagerNeeds

Enumeration	DtcKindEnum
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds
Note	This enumeration defines the possible kinds of diagnostic monitors regarding the OBD relevance.
Literal	Description
emission RelatedDtc	This indicates that the monitor reports a OBD-relevant malfunction.

nonEmmissionRelatedDtc	This indicates that the monitor reports a non-OBD-relevant malfunction.

Table 7.70: DtcKindEnum

The `diagEventDebounceAlgorithm` attribute defines the kind of expected debouncing by the Diagnostic Event Manager or defines that the debouncing is implemented by the software component.

The class `DiagEventDebounceAlgorithm` inherits from `Identifiable` in order to allow further documentation of the debouncing algorithm as well as non formalized description or non standardized description by the means of `Sdg` on expected configuration of the `DiagEventDebounceAlgorithm` in the Diagnostic Event Manager.

[constr_1138] `assignedPort` and `DiagEventDebounceMonitorInternal` [The existence of an `assignedPort` in combination with a `DiagEventDebounceAlgorithm` shall only be respected for the concrete subclass `DiagEventDebounceMonitorInternal`.]()

[constr_1139] `assignedPort` of `DiagEventDebounceMonitorInternal` shall refer to an `RPortPrototype` [Concerning the debouncing, the software-component acts as a client and thus the `assignedPort` defined with respect to a `DiagEventDebounceMonitorInternal` may only refer to an `RPortPrototype`. The standardized value of the `role` identifier of the `assignedPort` shall be `DiagFaultDetectionCounterPort`.]()

Class	DiagnosticEventNeeds			
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
Note	Specifies the abstract needs on the configuration of the Diagnostic Event Manager for one diagnostic event. Its <code>shortName</code> can be regarded as a symbol identifying the diagnostic event from the viewpoint of the component or module which owns this element. In case the diagnostic event specifies a production error, the <code>shortName</code> shall be the name of the production error.			
Base	ARObject, DiagnosticCapabilityElement , Identifiable , Multilanguage Referrable, Referrable , ServiceNeeds			
Attribute	Datatype	Mul.	Kind	Note
<code>considerPtoStatus</code>	Boolean	0..1	attr	PTO (Power Take Off) has an impact on the respective emission-related event (OBD). This information shall be provided by SW-C description in order to consider the PTO relevance e.g. for readiness (PID \$01) computation. For events with <code>dtcKind</code> set to 'nonEmmissionRelatedDtc' this attribute is typically false.
<code>deferringFim</code>	FunctionInhibitionNeeds	*	ref	This reference contains the link to a function identifier within the FiM which is used by the monitor before delivering a result.

Attribute	Datatype	Mul.	Kind	Note
diagEventDebounceAlgorithm	DiagEventDebounceAlgorithm	0..1	aggr	Specifies the abstract need on the Debounce Algorithm applied by the Diagnostic Event Manager.
dtcKind	DtcKindEnum	0..1	attr	This attribute indicates the kind of the diagnostic monitor according to the SWS Diagnostic Event Manager. This attribute applies for the UDS diagnostics use case.
inhibitingFid	FunctionInhibitionNeeds	0..1	ref	This represents the primary Function Inhibition Identifier used for inhibition of the diagnostic monitor. The FID might either inhibit the monitoring of a symptom or the reporting of detected faults.
inhibitingSecondaryFid	FunctionInhibitionNeeds	*	ref	This represents the secondary Function Inhibition Identifier used for inhibition of the diagnostic monitor. Any of the FID inhibitions leads to an inhibition of the monitoring of a symptom or the reporting of detected faults.
obdDtcNumber	PositiveInteger	0..1	attr	This represents a reasonable Diagnostic Trouble Code. This allows to predefine the Diagnostic Trouble Code, e.g. if the a function developer has received a particular requirement from the OEM or from a standardization body. This attribute applies for the OBD diagnostics use case.
reportBehavior	ReportBehaviorEnum	0..1	attr	This switch indicates whether or not the BSW module is allowed to report the related Events before Dem_Init().
udsDtcNumber	PositiveInteger	0..1	attr	This represents a reasonable Diagnostic Trouble Code. This allows to predefine the Diagnostic Trouble Code, e.g. if the a function developer has received a particular requirement from the OEM or from a standardization body. This attribute applies for the UDS diagnostics use case.

Table 7.71: DiagnosticEventNeeds

Class	DiagEventDebounceAlgorithm (abstract)			
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
Note	This class represents the ability to specify the pre-debounce algorithm which is selected and/or required by the particular monitor. This class inherits from Identifiable in order to allow further documentation of the expected or implemented debouncing and to use the category for the identification of the expected / implemented debouncing.			
Base	ARObject, Identifiable , MultilanguageReferrable, Referrable			
Attribute	Datatype	Mul.	Kind	Note

Attribute	Datatype	Mul.	Kind	Note
—	—	—	—	—

Table 7.72: DiagEventDebounceAlgorithm

Class	DiagEventDebounceCounterBased			
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
Note	This meta-class represents the ability to indicate that the counter-based debounce algorithm shall be used by the DEM for this diagnostic monitor. This is related to set the ECUC choice container DemDebounceAlgorithmClass to DemDebounceCounterBased.			
Base	ARObject, DiagEventDebounceAlgorithm , Identifiable , Multilanguage Referrable, Referrable			
Attribute	Datatype	Mul.	Kind	Note
counterDecrementStepSize	Integer	1	attr	This value shall be taken to decrement the internal debounce counter.
counterFailedThreshold	Integer	1	attr	This value defines the event-specific limit that indicates the "failed" counter status.
counterIncrementStepSize	Integer	1	attr	This value shall be taken to increment the internal debounce counter.
counterJumpDown	Boolean	1	attr	This value activates or deactivates the counter jump-down behavior.
counterJumpDownValue	Integer	1	attr	This value represents the initial value of the internal debounce counter if the counting direction changes from incrementing to decrementing.
counterJumpUp	Boolean	1	attr	This value activates or deactivates the counter jump-up behavior.
counterJumpUpValue	Integer	1	attr	This value represents the initial value of the internal debounce counter if the counting direction changes from decrementing to incrementing.
counterPassedThreshold	Integer	1	attr	This value defines the event-specific limit that indicates the "passed" counter status.

Table 7.73: DiagEventDebounceCounterBased

Class	DiagEventDebounceTimeBased			
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
Note	This meta-class represents the ability to indicate that the time-based pre-debounce algorithm shall be used by the DEM for this diagnostic monitor. This is related to set the ECUC choice container DemDebounceAlgorithmClass to DemDebounceTimeBase.			
Base	ARObject, DiagEventDebounceAlgorithm , Identifiable , Multilanguage Referrable, Referrable			
Attribute	Datatype	Mul.	Kind	Note
timeFailedThreshold	TimeValue	1	attr	This value represents the event-specific delay indicating the "failed" status.
timePassedThreshold	TimeValue	1	attr	This value represents the event-specific delay indicating the "passed" status.

Table 7.74: DiagEventDebounceTimeBased

Class	DiagEventDebounceMonitorInternal			
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
Note	This meta-class represents the ability to indicate that the pre-debounce algorithm shall be used by the DEM for this diagnostic monitor. This is related to setting the ECUC choice container DemDebounceAlgorithmClass to DemDebounceMonitorInternal. If the FaultDetectionAlgorithm is already known to be implemented by a specific BswModuleEntry the reference bswModuleEntry points to the function specification. If the FaultDetectionCounter value is accessible at a PortPrototype this PortPrototype shall be referenced by an assignedPort.			
Base	ARObject, DiagEventDebounceAlgorithm , Identifiable , Multilanguage Referrable, Referrable			
Attribute	Datatype	Mul.	Kind	Note
—	—	—	—	—

Table 7.75: DiagEventDebounceMonitorInternal

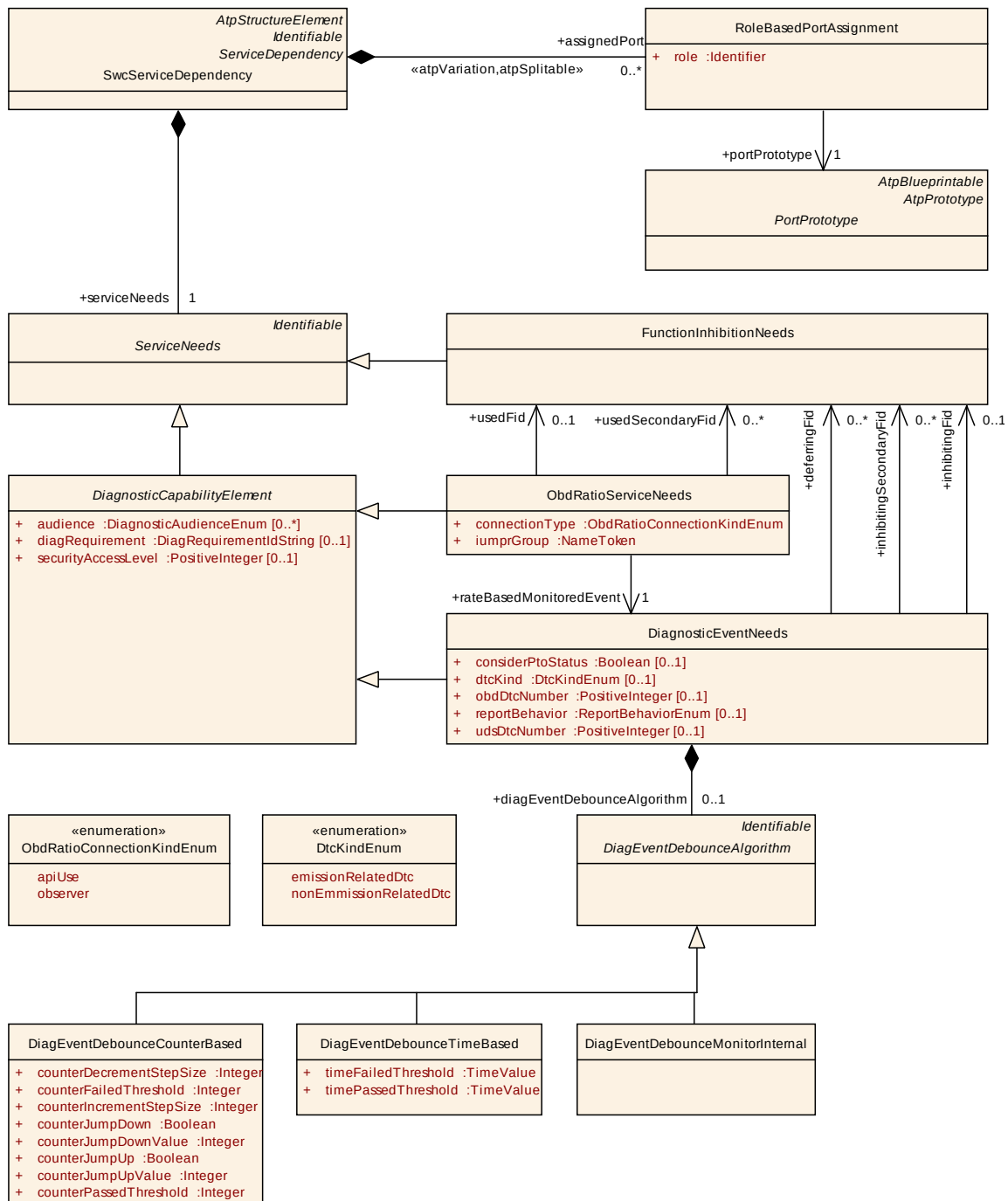


Figure 7.45: Relationship of [DiagnosticEventNeeds](#) and [FunctionInhibitionNeeds](#)

The figure 7.45 shows the relationship of the class [DiagnosticEventNeeds](#). The given M2 structure support to express following properties of a diagnostic monitor in addition to the basic set of attributes provided by [DiagnosticCapabilityElement](#):

With the [inhibitingFid](#) reference to an [FunctionInhibitionNeeds](#) instance on M1 it is declared that either the monitoring of a symptom or the reporting of detected faults can be inhibited by the usage of the Function Inhibition Managers.

The used [PortPrototype](#) which has to be connected to the Function Inhibition Managers is determined by the [RoleBasedPortAssignment](#) of the related [FunctionInhibitionNeeds](#) instance on M1.

The reference from a M1 instance of an [ObdRatioServiceNeeds](#) to an M1 instance of a [DiagnosticEventNeeds](#) specifies that the related Diagnostic Monitor supports Rate Based Monitoring. For further details see [7.11.3.7.4](#)

[TPS_SWCT_01582] Semantics of [DiagnosticEventNeeds.deferringFid](#) [Diagnostic monitor implementations use *Function Identifiers* (FID) to acquire permission from `FIM` before executing the fault detection.

Typically, the permission is not granted by `FIM` if other *Events* have already been reported as *FAILED*, which would lead to a double-detection of the same failure.

In some cases (see [35]), diagnostic monitor implementations do not only shut down completely in case of “no permission”, but fully compute their result and do just not deliver it to `Dem` before further conditions are fulfilled.

Typically, such diagnostics can detect a coarse failure quickly. But it avoids reporting *FAIL* early to give other Events a chance to deliver a more precise *FAIL*.

In such cases, the delivery of the result is only allowed when `FIM` grants a permission, with inhibitions on `NOT_TESTED` of other Events. These *Function Inhibitions* are specified by means of the attribute [DiagnosticEventNeeds.deferringFid](#).

]([RS_SWCT_00170](#), [RS_SWCT_03190](#))

As a corresponding concept to [DiagnosticEventNeeds](#), the [DiagnosticEventInfoNeeds](#) represents the needs to a given software-component that is interested to get information about specific DTCs.

Class	DiagnosticEventInfoNeeds			
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
Note	This meta-class represents the needs of a software-component interested to get information regarding specific DTCs.			
Base	ARObject, DiagnosticCapabilityElement , Identifiable ,Multilanguage Referrable, Referrable , ServiceNeeds			
Attribute	Datatype	Mul.	Kind	Note
dtcKind	DtcKindEnum	0..1	attr	This attribute indicates the kind of the diagnostic event according to the SWS Diagnostic Event Manger for which the DiagnosticInfo is requested. This attribute applies for the UDS diagnostics use case.

Attribute	Datatype	Mul.	Kind	Note
obdDtcNumber	PositiveInteger	0..1	attr	This represents a reasonable Diagnostic Trouble Code. This allows to predefine the Diagnostic Trouble Code, e.g. if the function developer has received a particular requirement from the OEM or from a standardization body. This attribute applies for the OBD diagnostics use case.
udsDtcNumber	PositiveInteger	0..1	attr	This represents a reasonable Diagnostic Trouble Code. This allows to predefine the Diagnostic Trouble Code, e.g. if the function developer has received a particular requirement from the OEM or from a standardization body. This attribute applies for the UDS diagnostics use case.

Table 7.76: DiagnosticEventInfoNeeds

Class	DiagnosticOperationCycleNeeds			
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
Note	This meta-class represents the needs of a software-component to provide information regarding the operation cycle management to the Dem module.			
Base	ARObject, DiagnosticCapabilityElement , Identifiable , Multilanguage Referrable, Referrable , ServiceNeeds			
Attribute	Datatype	Mul.	Kind	Note
operationCycle	OperationCycleTypeEnum	1	attr	Operation cycles types for the Dem to be supported by cycle-state APIs.
operationCycleAutomaticEnd	Boolean	1	attr	If this attribute is set to true the Dem shall automatically end the driving cycle at either Dem_Shutdown() or Dem_Init().
operationCycleAutostart	Boolean	1	attr	If this attribute is set to true the operation cycles is automatically (re-)started during Dem_PreInit().

Table 7.77: DiagnosticOperationCycleNeeds

Enumeration	OperationCycleTypeEnum
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds
Note	The possible values of the operation cycles types for the Dem.
Literal	Description
ignition	Ignition ON / OFF cycle.
obdDcy	OBD Driving cycle.
other	Further operation cycle.
power	Power ON / OFF cycle.

time	Time based operation cycle.
warmup	OBD Warm up cycle.

Table 7.78: OperationCycleTypeEnum

Class	DiagnosticEnableConditionNeeds			
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
Note	This meta-class represents the needs of a software-component to provide the capability to set an enable condition.			
Base	ARObject, DiagnosticCapabilityElement , Identifiable , Multilanguage Referrable, Referrable , ServiceNeeds			
Attribute	Datatype	Mul.	Kind	Note
initialStatus	EventAcceptanceStatusEnum	1	attr	Defines the initial status for enable or disable of acceptance of event reports of a diagnostic event.

Table 7.79: DiagnosticEnableConditionNeeds

Enumeration	EventAcceptanceStatusEnum			
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
Note	This enumerator specifies the initial status for enable or disable of acceptance of event reports of a diagnostic event.			
Literal	Description			
eventAcceptanceDisabled	Acceptance of a diagnostic event is disabled.			
eventAcceptanceEnabled	Acceptance of a diagnostic event is enabled.			

Table 7.80: EventAcceptanceStatusEnum

Class	DiagnosticStorageConditionNeeds			
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
Note	This meta-class represents the needs of a software-component to provide the capability to set a storage condition.			
Base	ARObject, DiagnosticCapabilityElement , Identifiable , Multilanguage Referrable, Referrable , ServiceNeeds			
Attribute	Datatype	Mul.	Kind	Note
initialStatus	StorageConditionStatusEnum	1	attr	Defines the initial status for enable or disable of storage of a diagnostic event.

Table 7.81: DiagnosticStorageConditionNeeds

Enumeration	StorageConditionStatusEnum			
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
Note	This enumeration specifies the initial status for enable or disable of storage of a diagnostic event.			

<i>Literal</i>	<i>Description</i>
eventStorage Disabled	Storage of a diagnostic event is disabled.
eventStorage Enabled	Storage of a diagnostic event is enabled.

Table 7.82: StorageConditionStatusEnum

Class	DtcStatusChangeNotificationNeeds			
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
Note	This meta-class represents the needs of a software-component interested to get information regarding any DTC status change.			
Base	ARObject, DiagnosticCapabilityElement , Identifiable , Multilanguage Referrable, Referrable , ServiceNeeds			
Attribute	Datatype	Mul.	Kind	Note
dtcFormat Type	DtcFormatTypeEnum	1	attr	This attribute specifies the DTC format.

Table 7.83: DtcStatusChangeNotificationNeeds

Enumeration	DtcFormatTypeEnum
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds
Note	This enumeration specifies the DTC format.
Literal	Description
j1939	Defines the J1939 DTC format.
obd	Defines the OBD DTC format.
uds	Defines the UDS DTC format.

Table 7.84: DtcFormatTypeEnum

7.11.3.7.2.1 Dem Service Use Case: diagnostic monitor, debouncing by Dem

Scenario: an [AtomicSwComponentType](#) implements a Diagnostic Monitor. The debouncing of the failure condition shall be configured and processed by the Dem. In this case the following setup apply:

[TPS_SWCT_01028] [AtomicSwComponentType](#) implements a Diagnostic Monitor [

ServiceNeeds kind [DiagnosticEventNeeds](#)

RoleBasedPortAssignment valid roles:

- DiagnosticMonitor [1]
- DiagnosticInfo [0 .. 1]
- CallbackInitMonitorForEvent [0 .. 1]

- CallbackEventStatusChange [0 .. 1]
- CallbackClearEventAllowed [0 .. 1]

RoleBasedDataAssignment

N/A

RepresentedPortGroups

N/A

]([RS_SWCT_00170](#), [RS_SWCT_03190](#))

Please note that for the implementation of this scenario [DiagEventDebounceCounterBased](#) or [DiagEventDebounceTimeBased](#) algorithm should be used as [diagEventDebounceAlgorithm](#).

7.11.3.7.2.2 Dem Service Use Case: diagnostic monitor, debouncing by SWC

Scenario: an [AtomicSwComponentType](#) implements a Diagnostic Monitor. The debouncing of the failure condition shall be processed by the software component. In this case the following setup applies:

[TPS_SWCT_01029] [AtomicSwComponentType](#) implements a Diagnostic Monitor [

ServiceNeeds kind [DiagnosticEventNeeds](#)

RoleBasedPortAssignment valid roles:

- DiagnosticMonitor [1]
- DiagnosticInfo [0 .. 1]
- CallbackInitMonitorForEvent [0 .. 1]
- CallbackEventStatusChange [0 .. 1]
- CallbackClearEventAllowed [0 .. 1]
- CallbackGetFaultDetectCounter [1]

RoleBasedDataAssignment

N/A

RepresentedPortGroups

N/A

]([RS_SWCT_00170](#), [RS_SWCT_03190](#))

Please note that for the implementation of this scenario [DiagEventDebounceMonitorInternal](#) algorithm should be used as [diagEventDebounceAlgorithm](#).

7.11.3.7.2.3 Dem Service Use Case: software-component provides information about operation cycles

Scenario: an [AtomicSwComponentType](#) provides information about operating cycles, e.g. ignition cycle or driving cycle.

[TPS_SWCT_01132] [AtomicSwComponentType](#) provides information about operating cycles [

ServiceNeeds kind [DiagnosticOperationCycleNeeds](#)

RoleBasedPortAssignment valid roles:

- [OperationCycle](#) [1]

RoleBasedDataAssignment

N/A

RepresentedPortGroups

N/A

]([RS_SWCT_00170](#), [RS_SWCT_03190](#))

For more information please refer to [SWS_Dem_00601] and [ECUC_Dem_00703].

7.11.3.7.2.4 Dem Service Use Case: software-component provides information about aging cycles

Scenario: an [AtomicSwComponentType](#) provides information about aging cycles.

[TPS_SWCT_01133] [AtomicSwComponentType](#) provides information about aging cycles [

ServiceNeeds kind [DiagnosticEventManagerNeeds](#)

RoleBasedPortAssignment valid roles:

- [AgingCycle](#) [0 .. 1]

RoleBasedDataAssignment

N/A

RepresentedPortGroups

N/A

]([RS_SWCT_00170](#), [RS_SWCT_03190](#))

For more information please refer to [SWS_Dem_00602].

7.11.3.7.2.5 Dem Service Use Case: software-component enables storage of DTCs in general

Scenario: a [AtomicSwComponentType](#) enables the storage of DTCs in general.

[TPS_SWCT_01134] [AtomicSwComponentType](#) enables storage of DTCs in general [

ServiceNeeds kind [DiagnosticEnableConditionNeeds](#)

RoleBasedPortAssignment valid roles:

- [EnableCondition](#) [1]

RoleBasedDataAssignment

N/A

RepresentedPortGroups

N/A

]([RS_SWCT_00170](#), [RS_SWCT_03190](#))

For more information please refer to [SWS_Dem_00604] and [ECUC_Dem_00656].

7.11.3.7.2.6 Dem Service Use Case: software-component enables storage of subsequent DTCs

Scenario: an [AtomicSwComponentType](#) enables the storage of subsequent DTCs.

[TPS_SWCT_01135] [AtomicSwComponentType](#) enables storage of subsequent DTCs [

ServiceNeeds kind [DiagnosticStorageConditionNeeds](#)

RoleBasedPortAssignment valid roles:

- [StorageCondition](#) [1]

RoleBasedDataAssignment

N/A

RepresentedPortGroups

N/A

]([RS_SWCT_00170](#), [RS_SWCT_03190](#))

For more information please refer to [SWS_Dem_00605].

The relevant DTCs shall be configured in ECUC because at the time the [AtomicSwComponentType](#) is designed the information about which DTCs are relevant is not fully available.

7.11.3.7.2.7 Dem Service Use Case: retrieve information of the lamp status

Please note that for this specific use case the application of a concrete [ServiceNeeds](#) is not yet clarified.

Scenario: an [AtomicSwComponentType](#) retrieves information of the lamp status.

[TPS_SWCT_01136] [AtomicSwComponentType](#) retrieves information of the lamp status [

RoleBasedPortAssignment valid roles:

- [IndicatorStatus](#) [1]

RoleBasedDataAssignment

N/A

RepresentedPortGroups

N/A

]([RS_SWCT_00170](#), [RS_SWCT_03190](#))

For more information please refer to [SWS_Dem_00606].

7.11.3.7.2.8 Dem Service Use Case: DEM provides information that the fault storage overflows

Please note that for this specific use case the application of a concrete [ServiceNeeds](#) is not yet clarified.

Scenario: the Dem provides information that the fault storage overflows.

[TPS_SWCT_01137] Dem provides information that the fault storage overflows [

RoleBasedPortAssignment valid roles:

- [EvMemOverflowIndication](#) [1]

RoleBasedDataAssignment

N/A

RepresentedPortGroups

N/A

]([RS_SWCT_00170](#), [RS_SWCT_03190](#))

For more information please refer to [SWS_Dem_00607].

7.11.3.7.2.9 Dem Service Use Case: software-component suppresses the storage of DTCs

Scenario: an [AtomicSwComponentType](#) suppresses the storage of DTCs within the Dem.

[TPS_SWCT_01138] [AtomicSwComponentType](#) suppresses the storage of DTCs within the Dem [

ServiceNeeds kind [DiagnosticEventManagerNeeds](#)

RoleBasedPortAssignment valid roles:

- [DTCsuppression](#) [1]

RoleBasedDataAssignment

N/A

RepresentedPortGroups

N/A

]([RS_SWCT_00170](#), [RS_SWCT_03190](#))

For more information please refer to [SWS_Dem_00608].

7.11.3.7.2.10 Dem Service Use Case: software-component informs that the PTO is active

Scenario: an [AtomicSwComponentType](#) informs the Dem that the PTO is active.

[TPS_SWCT_01139] [AtomicSwComponentType](#) informs the Dem that the PTO is active [

ServiceNeeds kind [DiagnosticEventManagerNeeds](#)

RoleBasedPortAssignment

The following roles are applicable:

- [PowerTakeOff](#) [1]

RoleBasedDataAssignment

N/A

RepresentedPortGroups

N/A

]([RS_SWCT_00170](#), [RS_SWCT_03190](#))

For more information please refer to [SWS_Dem_00612].

7.11.3.7.2.11 Dem Service Use Case: software-component needs information about any DTC status change

Scenario: an `AtomicSwComponentType` needs information about any DTC status change. There is no limitation on the number of software-components requesting the information.

[TPS_SWCT_01140] `AtomicSwComponentType` needs information about specific DTC without being a diagnostic monitor [

ServiceNeeds kind `DtcStatusChangeNotificationNeeds`

RoleBasedPortAssignment valid roles:

- `CallbackDTCStatusChange` [1]

RoleBasedDataAssignment

N/A

RepresentedPortGroups

N/A

](`RS_SWCT_00170`, `RS_SWCT_03190`)

For more information please refer to [SWS_Dem_00617].

In the case the software-component needs notifications about different kinds of the DTC status change (formalized by `DtcFormatTypeEnum`) it is applicable to create a `SwcServiceDependency` for each kind of status change.

7.11.3.7.2.12 Dem Service Use Case: call operation if the data of a given diagnostic event changes (I)

Scenario: an `AtomicSwComponentType` provides a `PPortPrototype` typed by the `ClientServerInterface` `CallbackEventDataChanged`. The service component calls the `ClientServerOperation` `EventDataChanged` if the corresponding diagnostic event changes in terms of the underlying data.

For each diagnostic events to which the `AtomicSwComponentType` is conceptually connected it needs to provide one `PPortPrototype` towards the service component.

[TPS_SWCT_01425] `AtomicSwComponentType` provides one callback per event if diagnostic event data change [

ServiceNeeds kind `DiagnosticEventInfoNeeds`

RoleBasedPortAssignment valid roles:

- `CallbackEventDataChanged` [1]

RoleBasedDataAssignment

N/A

RepresentedPortGroups

N/A

|(RS_SWCT_00170, RS_SWCT_03190)

For more information please refer to [SWS_Dem_00618].

7.11.3.7.2.13 Dem Service Use Case: call operation if the data or status of any diagnostic event changes (II)

Scenario: an [AtomicSwComponentType](#) shall react on any diagnostic event status change and/or any diagnostic event data change. For instance this may be used to write a time stamp when any event status changes regardless of the event id.

In contrast to the scenario described in chapter [7.11.3.7.2.12](#) or [7.11.3.7.2.11](#) this case foresees the existence of a single [PPortPrototype](#) that covers all relevant diagnostic events.

[TPS_SWCT_01426] [AtomicSwComponentType](#) provides callback if any diagnostic event data and/or status changed [

ServiceNeeds kind [DiagnosticEventManagerNeeds](#)

RoleBasedPortAssignment valid roles:

- [GeneralCallbackEventDataChanged](#) [0..1]
- [GeneralCallbackEventStatusChange](#) [0..1]
- [GeneralDiagnosticInfo](#) [0..1]

RoleBasedDataAssignment

N/A

RepresentedPortGroups

N/A

For more information please refer to [SWS_Dem_00616], [SWS_Dem_00619], and [SWS_Dem_00600].

In order to react on diagnostic event status changes the software component shall provide a single [PPortPrototype](#) typed as a client server interface compatible to [GeneralCallbackEventDataChanged](#).

In order to react on diagnostic event data changes the software component shall provide a single [PPortPrototype](#) typed as a client server interface compatible to [GeneralCallbackEventDataChanged](#).

If the software-component additionally has to read further information of the specific diagnostic event from Dem it shall provide a [RPortPrototype](#) typed as a client server interface compatible to [GeneralDiagnosticInfo](#). It shall also specify [DiagnosticEventInfoNeeds](#). |(RS_SWCT_00170, RS_SWCT_03190)

7.11.3.7.2.14 Dem Service Use Case: software-component provides data for diagnostic purposes

Please note that for this specific use case the application of a concrete [ServiceNeeds](#) is not yet clarified.

Scenario: an [AtomicSwComponentType](#) provides data to be used for diagnostic purposes. The provision of data can be done by means of [PPortPrototypes](#) typed by either [ClientServerInterfaces](#) or [SenderReceiverInterfaces](#). The usage of the latter, however, is not further detailed in the applicable SWS [36] and therefore no more details are to be provided in this document.

[TPS_SWCT_01427] [AtomicSwComponentType](#) provides data for diagnostic purposes via [ClientServerInterface](#) [

RoleBasedPortAssignment valid roles:

- [DataServices](#) [1]

RoleBasedDataAssignment

N/A

RepresentedPortGroups

N/A

]([RS_SWCT_00170](#), [RS_SWCT_03190](#))

[TPS_SWCT_01634] Suffix used for the resulting name of the [PortInterface](#) for the Data Services [The *suffix* used for the resulting name of the [PortInterface](#) for the Data Services ([DataServices_{Data}](#)) shall be taken from the [shortName](#) of the applicable [SwcServiceDependency](#). **]([RS_SWCT_00170](#), [RS_SWCT_03190](#))**

For more information please refer to [SWS_Dem_00621].

7.11.3.7.2.15 Dem Service Use Case: interface to DCM

Please note that for this specific use case the application of a concrete [ServiceNeeds](#) is not yet clarified.

Scenario: a [ServiceSwComponentType](#) representing the Dem provides a [PPortPrototype](#) for the Dcm. Although this scenario does not apply to [ApplicationSwComponentTypes](#) it is included for the sake of completeness.

[TPS_SWCT_01428] [ServiceSwComponentType](#) representing the Dem provides a [PPortPrototype](#) for the Dcm [

RoleBasedPortAssignment valid roles:

- [DcmIf](#) [1]

RoleBasedDataAssignment

N/A

RepresentedPortGroups

N/A

]([RS_SWCT_00170](#), [RS_SWCT_03190](#))

For more information please refer to [SWS_Dem_00609].

7.11.3.7.2.16 Dem Service Use Case: software-component gets information about a specific DTC

Scenario: an [AtomicSwComponentType](#) specifies [DiagnosticEventInfoNeeds](#) in order to be able to get information about specific DTCs. This use case to some extent is similar to [[TPS_SWCT_01426](#)] but does not replace that use case.

[TPS_SWCT_01453] Software-component gets information about a specific DTC

[

ServiceNeeds kind [DiagnosticEventInfoNeeds](#)

RoleBasedPortAssignment valid roles:

- [DiagnosticInfo](#) [1]

RoleBasedDataAssignment

N/A

RepresentedPortGroups

N/A

]([RS_SWCT_00170](#), [RS_SWCT_03190](#))

For more information please refer to [SWS_Dem_00609].

7.11.3.7.3 Diagnostic Communication Needs

The meta-class [DiagnosticCommunicationManagerNeeds](#) is used to define requirements in order to configure the Diagnostic Communication Manager Service.

An [SwcInternalBehavior](#) may provide a [DiagnosticCommunicationManagerNeeds](#) element which defines the mappings for the general diagnostic communication (for the terms related to the AUTOSAR Diagnostic Communication Manager see [37]).

Class	DiagnosticCommunicationManagerNeeds			
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
Note	Specifies the general needs on the configuration of the Diagnostic Communication Manager (DCM) which are not related to a particular item (e.g. a PID or DiagnosticRoutineNeeds). The main use case is the mapping of service ports to the DCM which are not related to a particular item.			
Base	ARObject, DiagnosticCapabilityElement , Identifiable , Multilanguage Referrable, Referrable , ServiceNeeds			
Attribute	Datatype	Mul.	Kind	Note
serviceRequestCallbackType	DiagnosticServiceRequestCallbackTypeEnum	0..1	attr	This represents the ability to define whether the usage of PortInterface ServiceRequestNotification has the characteristics of being initiated by a manufacturer or by a supplier.

Table 7.85: DiagnosticCommunicationManagerNeeds

Enumeration	DiagnosticServiceRequestCallbackTypeEnum			
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
Note	This represents the ability to define whether a Service Request Notification was used in the role of a manufacturer or a supplier.			
Literal	Description			
requestCallbackTypeManufacturer	This represents the case that the usage of PortInterface ServiceRequestNotification has the characteristics of being used by a manufacturer.			
requestCallbackTypeSupplier	This represents the case that the usage of PortInterface ServiceRequestNotification has the characteristics of being used by a supplier.			

Table 7.86: DiagnosticServiceRequestCallbackTypeEnum

The meta-class [DiagnosticRoutineNeeds](#) is used to define requirements to configure the Diagnostic Communication Manager Service. A [PPortPrototype](#) typed by a [ClientServerInterface](#)⁷ may provide [ClientServerOperations](#) (for example, “start”, “stop”, and “RequestResults”).

The [PPortPrototype](#) corresponds to the diagnostic service `RoutineControl`. Within the [SwcInternalBehavior](#) up to three [RunnableEntity](#)s are defined for implementing the [ClientServerOperations](#) mentioned before.

The enumeration parameter [DiagnosticRoutineTypeEnum](#) is used to define whether the diagnostic server or client is responsible for stopping the routine.

Please note that [constr_1340] and [constr_1341] apply for the application of [DiagnosticRoutineNeeds](#). These constraints are part of the specification of the [DiagnosticExtract](#) [38].

⁷where `isService` shall be set to true

Class	DiagnosticRoutineNeeds			
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
Note	Specifies the general needs on the configuration of the Diagnostic Communication Manager (DCM) which are not related to a particular item (e.g. a PID). The main use case is the mapping of service ports to the DCM which are not related to a particular item.			
Base	ARObject, DiagnosticCapabilityElement , Identifiable , Multilanguage Referrable, Referrable , ServiceNeeds			
Attribute	Datatype	Mul.	Kind	Note
diagRoutineType	DiagnosticRoutineTypeEnum	1	attr	This denotes the type of diagnostic routine which is implemented by the referenced server port.
ridNumber	PositiveInteger	0..1	attr	This represents a routine identifier for the diagnostic routine. This allows to predefine the RID number if the a function developer has received a particular requirement from the OEM or from a standardization body.

Table 7.87: DiagnosticRoutineNeeds

Enumeration	DiagnosticRoutineTypeEnum
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds
Note	This enumerator specifies the different types of diagnostic routines.
Literal	Description
asynchronous	This indicates that the diagnostic server is not blocked while the diagnostic routine is running.
synchronous	This indicates that the diagnostic routine blocks the diagnostic server in the ECU while the routine is running.

Table 7.88: DiagnosticRoutineTypeEnum

The meta-class [DiagnosticIoControlNeeds](#) is used to define requirements to configure the Diagnostic Communication Manager Service. The [PPortPrototype](#) corresponds to the diagnostic service [InputOutputControlByIdentifier](#). Within the [SwcInternalBehavior](#) up to three [RunnableEntities](#) are defined for implementing the [ClientServerOperations](#) mentioned before.

Class	DiagnosticIoControlNeeds			
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
Note	Specifies the general needs on the configuration of the Diagnostic Communication Manager (DCM) which are not related to a particular item (e.g. a PID). The main use case is the mapping of service ports to the DCM which are not related to a particular item.			
Base	ARObject, DiagnosticCapabilityElement , Identifiable , Multilanguage Referrable, Referrable , ServiceNeeds			
Attribute	Datatype	Mul.	Kind	Note
currentValue	DiagnosticValueNeeds	0..1	ref	Reference to the DiagnosticValueNeeds indicating the access to the current value via signalBasedDiagnostics .

Attribute	Datatype	Mul.	Kind	Note
didNumber	PositiveInteger	0..1	attr	This represents a Data identifier for the diagnostic value. This allows to predefine the DID number if the a function developer has received a particular requirement from the OEM or from a standardization body.
freezeCurrentStateSupported	Boolean	0..1	attr	This attribute determines, if the referenced port supports temporary freezing of I/O value.
resetToDefaultSupported	Boolean	0..1	attr	This represents a flag for the existence of the ResetToDefault operation in the service interface.
shortTermAdjustmentSupported	Boolean	0..1	attr	This attribute determines, if the referenced port supports temporarily setting of I/O value to a specific value provided by the diagnostic tester.

Table 7.89: DiagnosticIoControlNeeds

The meta-class [DiagnosticValueNeeds](#) is used to define requirements in order to configure the Diagnostic Communication Manager Service as well as the Diagnostic Event Manager Service.

The DCM can access either local values via a [ClientServerInterface](#) or it may access [dataElements](#) in a [PPortPrototype](#) typed by a [SenderReceiverInterface](#). For this purpose, the [DiagnosticValueNeeds](#) require associations to local values (i.e. inside [InternalBehavior](#)) or respectively [dataElements](#).

The attribute [DiagnosticValueNeeds.diagnosticValueAccess](#) of type [DiagnosticValueAccessEnum](#) allows for distinguishing between current values to read diagnostic information (readOnly) and data elements which are additionally classified as configurable (readWrite).

[constr_1363] Existence of attributes of [DiagnosticValueNeeds](#) [if [DiagnosticValueNeeds](#) is aggregated by a [SwcServiceDependency](#) in the role [serviceNeeds](#) then the attributes

- [DiagnosticValueNeeds.diagnosticValueAccess](#)
- [DiagnosticValueNeeds.dataLength](#)

shall **not** exist. [|\(\)](#)

[constr_1364] Existence of attributes of [DiagnosticIoControlNeeds](#) [if [DiagnosticIoControlNeeds](#) is aggregated by a [SwcServiceDependency](#) in the role [serviceNeeds](#) then the attributes

- [DiagnosticIoControlNeeds.freezeCurrentStateSupported](#)
- [DiagnosticIoControlNeeds.shortTermAdjustmentSupported](#)

shall **not** exist. [|\(\)](#)

For all intents and purposes, the statement made by [\[constr_1363\]](#) and [\[constr_1364\]](#) boils down to the fact that these attributes can only be reasonably used in the context of a [BswServiceDependency](#).

Class	DiagnosticValueNeeds			
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
Note	Specifies the general needs on the configuration of the Diagnostic Communication Manager (DCM) which are not related to a particular item (e.g. a PID). The main use case is the mapping of service ports to the DCM which are not related to a particular item. In the case of using a sender receiver communicated value, the related value shall be taken via assignedData in the role "signalBasedDiagnostics". In case of using a client/server communicated value, the related value shall be communicated via the port referenced by assignedPort. The details of this communication (e.g. appropriate naming conventions) are specified in the related software specifications (SWS).			
Base	ARObject, DiagnosticCapabilityElement , Identifiable , Multilanguage Referrable, Referrable , ServiceNeeds			
Attribute	Datatype	Mul.	Kind	Note
dataLength	PositiveInteger	0..1	attr	This attribute is applicable only if the ServiceNeed is aggregated within BswModuleDependency. This attribute represents the length of data (in bytes) provided for this particular PID signal.
diagnosticValueAccess	DiagnosticValueAccessEnum	0..1	attr	This attribute controls whether the data can be read and written or whether it is to be handled read-only.
didNumber	PositiveInteger	0..1	attr	This represents a Data identifier for the diagnostic value. This allows to predefine the DID number if the responsible function developer has received a particular requirement from the OEM or from a standardization body.
fixedLength	Boolean	0..1	attr	This attribute controls whether the data length of the data is fixed.
processingStyle	DiagnosticProcessingStyleEnum	0..1	attr	This attribute controls whether interaction requires the software-component to react synchronously on a request or whether it processes the request in background but still the DCM has to issue the call again to eventually obtain the result of the request.

Table 7.90: DiagnosticValueNeeds

Enumeration	DiagnosticValueAccessEnum
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds
Note	Defines the access of the configured diagnostic current values which will be used by the DEM or DCM module.
Literal	Description
readOnly	The access to the data element is limited to read-only. This is typically used to read-out diagnostic information (e.g. current values).

readWrite	The value of the diagnostic data element is classified as configurable (read and write access is possible).
writeOnly	The access to the data element is limited to write-only. This supports the use case where the Dcm just writes data to the application software without the intention to read it back,

Table 7.91: DiagnosticValueAccessEnum

Enumeration	DiagnosticProcessingStyleEnum
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds
Note	
Literal	Description
processingStyleAsynchronous	The software-component processes the request in background but still the DCM has to issue the call again to eventually obtain the result of the request.
processingStyleSynchronous	The software-component is superposed to react synchronously on the request.

Table 7.92: DiagnosticProcessingStyleEnum

Class	DiagnosticsCommunicationSecurityNeeds			
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
Note	This meta-class represents the needs of a software-component to verify the access to security level via diagnostic services.			
Base	ARObject, DiagnosticCapabilityElement , Identifiable , Multilanguage Referrable , Referrable , ServiceNeeds			
Attribute	Datatype	Mul.	Kind	Note
—	—	—	—	—

Table 7.93: DiagnosticsCommunicationSecurityNeeds

7.11.3.7.3.1 Dcm Service Use Case: read/write current values by Client Server Interface

Scenario: an [AtomicSwComponentType](#) offers a [PPortPrototype](#) typed by [ClientServerInterface](#) to read/write current value via diagnostic services (e.g. measurements, variant coding)

[TPS_SWCT_02002] [AtomicSwComponentType](#) offers a [PPortPrototype](#) typed by [ClientServerInterface](#) to read/write current value via diagnostic services

ServiceNeeds kind [DiagnosticValueNeeds](#)

RoleBasedPortAssignment valid roles:

- DataServices [1]

RoleBasedDataAssignment

N/A

RepresentedPortGroups

N/A

](RS_SWCT_00170, RS_SWCT_03190)

[TPS_SWCT_01628] Suffix used for the resulting name of the PortInterface for the Data Services [The *suffix* used for the resulting name of the PortInterface for the Data Services (DataServices_{Data}) shall be taken from the *shortName* of the applicable SwcServiceDependency.](RS_SWCT_00170, RS_SWCT_03190)

For more information please refer to [SWS_Dcm_00686].

7.11.3.7.3.2 Dcm Service Use Case: read/write current values of specific DID by Client Server Interface

Scenario: an AtomicSwComponentType offers a PPortPrototype typed by ClientServerInterface to read/write current values via diagnostic services (e.g. measurements, variant coding) where the applicable DID is passed as an argument to the access functions. This use case applies mostly if the software-component provides the information related to more than one DID.

[TPS_SWCT_01639] AtomicSwComponentType offers a PPortPrototype typed by ClientServerInterface to read/write current value via diagnostic services where the applicable DID is passed as an argument to the access functions [

ServiceNeeds kind DiagnosticValueNeeds

RoleBasedPortAssignment valid roles:

- DataServices_DIDRange [1]

RoleBasedDataAssignment

N/A

RepresentedPortGroups

N/A

](RS_SWCT_00170, RS_SWCT_03190)

[TPS_SWCT_01640] Suffix used for the resulting name of the PortInterface for the Data Services [The *suffix* used for the resulting name of the PortInterface for the Data Services (DataServices_DIDRange_{Range}) shall be taken from the *shortName* of the applicable SwcServiceDependency.](RS_SWCT_00170, RS_SWCT_03190)

For more information please refer to [SWS_Dcm_00769].

7.11.3.7.3.3 Dcm Service Use Case: read/write current values by Sender Receiver Interface

Scenario: an [AtomicSwComponentType](#) offers [PortPrototypes](#) typed by [SenderReceiverInterfaces](#) to read/write current values via diagnostic services (e.g. measurements, variant coding) This is mainly used for data which are available at ports anyhow used for other communication purpose.

Note: this scenario can be implemented as a regular sender/receiver communication without the necessity to use a [SwcServiceDependency](#). The description of a [SwcServiceDependency](#) (even if it is technically not required) may help to advertise the special role of the corresponding [dataElement](#) with respect to diagnostics.

[TPS_SWCT_02003] [AtomicSwComponentType](#) offers [PortPrototypes](#) typed by [SenderReceiverInterfaces](#) to read/write current values via diagnostic services [

ServiceNeeds kind [DiagnosticValueNeeds](#)

RoleBasedPortAssignment

N/A

RoleBasedDataAssignment valid roles:

- [signalBasedDiagnostics](#) [1..2]

RepresentedPortGroups

N/A

To read the signal the [AtomicSwComponentType](#) shall offer an [AbstractProvidedPortPrototype](#), to write the signal the [AtomicSwComponentType](#) shall offer an [AbstractRequiredPortPrototype](#).]([RS_SWCT_00170](#), [RS_SWCT_03190](#))

For more information please refer to [SWS_Dcm_00687].

7.11.3.7.3.4 Dcm Service Use Case: start/stop or request routine results

Scenario: an [AtomicSwComponentType](#) offers a [PortPrototype](#) typed by a [ClientServerInterface](#) to start/stop or request routine results of diagnostic routines.

[TPS_SWCT_02004] [AtomicSwComponentType](#) offers a [PortPrototype](#) typed by a [ClientServerInterface](#) to start/stop or request routine results of diagnostic routines [

ServiceNeeds kind [DiagnosticRoutineNeeds](#)

RoleBasedPortAssignment valid roles:

- [RoutineServices](#) [1]

RoleBasedDataAssignment

N/A

RepresentedPortGroups

N/A

]([RS_SWCT_00170](#), [RS_SWCT_03190](#))

[TPS_SWCT_01632] Suffix used for the resulting name of the [PortInterface](#) for the Routine Services [The *suffix* used for the resulting name of the [PortInterface](#) for the Routine Services (RoutineServices_{RoutineName}) shall be taken from the *shortName* of the applicable [SwcServiceDependency](#).]([RS_SWCT_00170](#), [RS_SWCT_03190](#))

For more information please refer to [SWS_Dcm_00690].

7.11.3.7.3.5 Dcm Service Use Case: IO control by Client Server Interface

Scenario: an [AtomicSwComponentType](#) offers a [PortPrototype](#) typed by a [ClientServerInterface](#) to adjust the IO signal via diagnostic services.

[TPS_SWCT_02005] [AtomicSwComponentType](#) offers [PortPrototypes](#) typed by [ClientServerInterfaces](#) to adjust the IO signal via diagnostic services [

ServiceNeeds kind [DiagnosticIoControlNeeds](#)

RoleBasedPortAssignment valid roles:

- [DataServices](#) [1]

RoleBasedDataAssignment

N/A

RepresentedPortGroups

N/A

]([RS_SWCT_00170](#), [RS_SWCT_03190](#))

[TPS_SWCT_01629] Suffix used for the resulting name of the [PortInterface](#) for the Data Services [The *suffix* used for the resulting name of the [PortInterface](#) for the Data Services (DataServices_{Data}) shall be taken from the *shortName* of the applicable [SwcServiceDependency](#).]([RS_SWCT_00170](#), [RS_SWCT_03190](#))

For more information please refer to [SWS_Dcm_00686].

7.11.3.7.3.6 Dcm Service Use Case: IO control by Sender Receiver Interface

This use case represents an alternative to the the use case described in chapter 7.11.3.7.3.5, i.e. for the same purpose it is also possible to utilize a `SenderReceiverInterface`.

The essential idea behind the existence of I/O `PortPrototypes` typed by `SenderReceiverInterface` is the possibility to have a quick access to the `dataElements` currently under control.

Especially cases where access to `dataElements` is required from different partitions (for example in multi core systems) can benefit from this approach.

Scenario: an `AtomicSwComponentType` offers an `RPortPrototype` typed by a `SenderReceiverInterface` (in particular: `IOControlRequest`) to adjust the I/O signal via diagnostic services and offers a `PPortPrototype` typed by a `SenderReceiverInterface` (in particular: `IOControlResponse`) to provide the IO “operation response”.

In case of using `IOControlRequest` (which owns **three** `dataElements`) and `IOControlResponse` the whole `PortPrototype` is related to **exactly one** IO control and needs to be consistent.

Therefore, the usage of `RoleBasedPortAssignment` (instead of the `RoleBasedDataAssignment`, which would otherwise typically be used for a sender/receiver-based scenario) is required for avoiding modeling overhead.

[TPS_SWCT_01654] `AtomicSwComponentType` offers `PortPrototypes` typed by `SenderReceiverInterfaces` to adjust the IO signal via diagnostic services

[

ServiceNeeds kind `DiagnosticIoControlNeeds`

RoleBasedPortAssignment valid roles:

- `IOControlRequest` [1]
- `IOControlResponse` [1]

RoleBasedDataAssignment

N/A

RepresentedPortGroups

N/A

](`RS_SWCT_00170`, `RS_SWCT_03190`)

The `IOControl` service requires in its diagnostic response the current value of the IO-DID, which is identical to the current value represented by `DiagnosticValueNeeds` of the `ReadDataByIdentifier` response.

[TPS_SWCT_01655] Reference from `DiagnosticIoControlNeeds` to `DiagnosticValueNeeds` [In the scenario described by [TPS_SWCT_01654], the `Diagnos-`

`ticIoControlNeeds` shall reference the `DiagnosticValueNeeds` which relates to the access of the current value via diagnostic services (see [TPS_SWCT_02003]).
|(RS_SWCT_00170, RS_SWCT_03190)

[TPS_SWCT_01656] **Suffix used for the resulting name of the `PortInterface` for `IOControlRequest` and `IOControlResponse`** | The *suffix* used for the resulting name of the `PortInterface` for the `IOControlRequest_Data` and `IOControlResponse_Data` shall be taken from the *shortName* of the applicable `SwcServiceDependency`. |(RS_SWCT_00170, RS_SWCT_03190)

The service use case is visualized in Figure 7.46. The `SwComponentPrototype` contains two `SwcServiceDependency`s, one for the I/O Control, and one for the access of the `dataElement` with the *shortName* “IOx” by the Dcm.

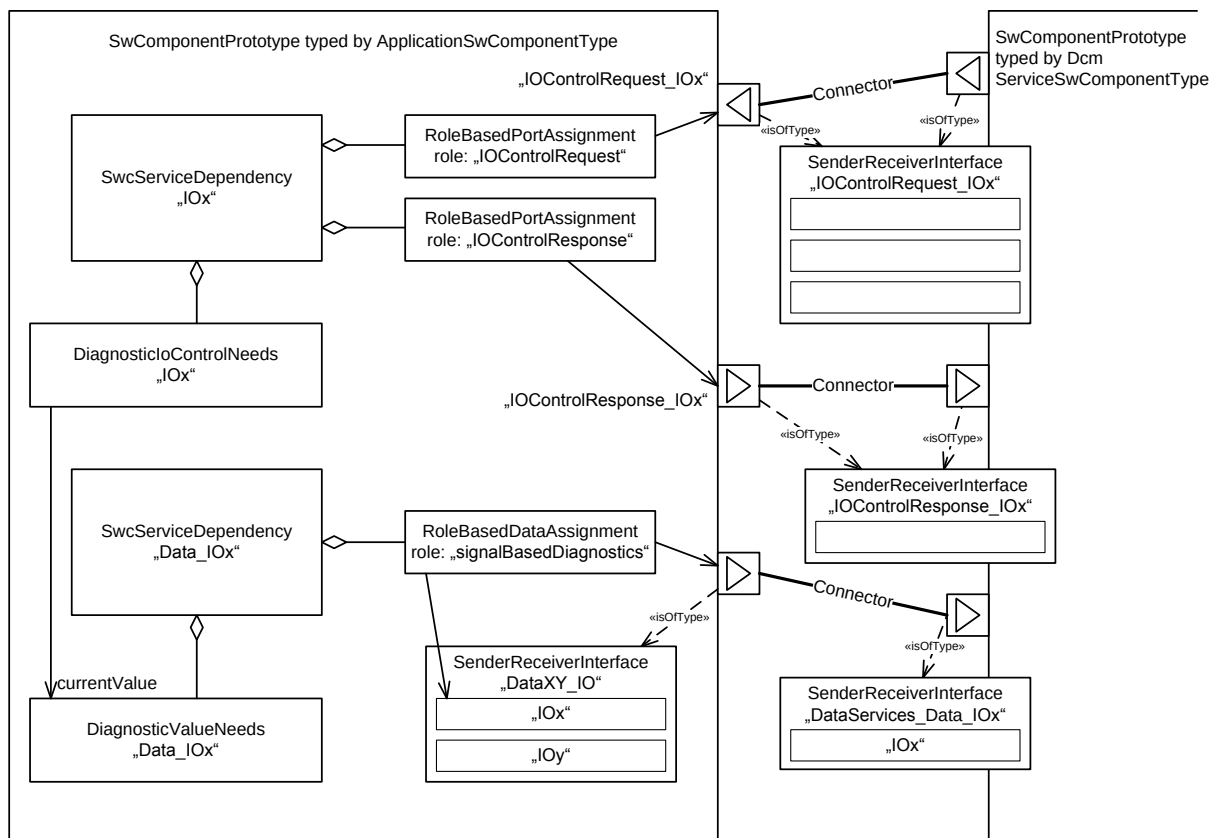


Figure 7.46: Visualization of the service use case

Please note that, in this example, the `SenderReceiverInterface` used on the `PPortPrototype` of the `ApplicationSwComponentType` has several `dataElements` (where the `dataElement` with the *shortName* “IOx” is one of them). This is a perfectly valid configuration.

On the other hand, the `SenderReceiverInterface` used on the `RPortPrototype` of the `ServiceSwComponentType` representing the Dcm can only have **one**

`dataElement`. This single `dataElement` shall (as far as the example is concerned) be given the `shortName` “IOx”.

Note the reference from the `DiagnosticIoControlNeeds` to the `DiagnosticValueNeeds`. this reference explicitly expresses that access to a DID is combined with the usage of I/O control.

[TPS_SWCT_01657] NamingRule for `RPortPrototype` referenced by a `RoleBasedPortAssignment` with attribute `role` set to “IOControlRequest” [The `shortName` of a `RPortPrototype` referenced by a `RoleBasedPortAssignment` with attribute `role` set to “IOControlRequest” shall be created by concatenating the prefix “IOControlRequest” and the `SwcServiceDependency.shortName`, separated by a single underscore character (i.e. “_”).]([RS_SWCT_00170](#), [RS_SWCT_03190](#))

[TPS_SWCT_01658] NamingRule for `PPortPrototype` referenced by a `RoleBasedPortAssignment` with attribute `role` set to “IOControlResponse” [The `shortName` of a `PPortPrototype` referenced by a `RoleBasedPortAssignment` with attribute `role` set to “IOControlResponse” shall be created by concatenating the prefix “IOControlResponse” and the `SwcServiceDependency.shortName`, separated by a single underscore character (i.e. “_”).]([RS_SWCT_00170](#), [RS_SWCT_03190](#))

For more information please refer to [SWS_Dcm_01308] and [SWS_Dcm_01309].

7.11.3.7.3.7 Dcm Service Use Case: Access to protocol, session and security information

Scenario: an `AtomicSwComponentType` offers a server port to get protocol, session and security information or to request a Reset to Default Session.

[TPS_SWCT_02013] `AtomicSwComponentType` offers a server port to get protocol, session and security information or to request a Reset to Default Session [

ServiceNeeds kind `DiagnosticCommunicationManagerNeeds`

RoleBasedPortAssignment valid roles:

- `DCMServices` [1]

RoleBasedDataAssignment
N/A

RepresentedPortGroups
N/A

] ([RS_SWCT_00170](#), [RS_SWCT_03190](#))

For more information please refer to [SWS_Dcm_00698]

7.11.3.7.3.8 Dcm Service Use Case: Verify the access to security level

Scenario: an [AtomicSwComponentType](#) provides a server port to verify the access to security level via diagnostic services.

[TPS_SWCT_02015] [AtomicSwComponentType](#) verifies the access to security level via diagnostic services [

ServiceNeeds kind [DiagnosticsCommunicationSecurityNeeds](#)

RoleBasedPortAssignment valid roles:

- [SecurityAccess](#) [1]

RoleBasedDataAssignment

N/A

RepresentedPortGroups

N/A

]([RS_SWCT_00170](#), [RS_SWCT_03190](#))

[TPS_SWCT_01627] Suffix used for the resulting name of the [PortInterface](#) for the Security Access [The *suffix* used for the resulting name of the [PortInterface](#) for the Security Access ([SecurityAccess_{SecurityLevel}](#)) shall be taken from the *shortName* of the applicable [SwcServiceDependency](#).]([RS_SWCT_00170](#), [RS_SWCT_03190](#))

For more information please refer to [SWS_Dcm_00685]

7.11.3.7.3.9 Dcm Service Use Case: multiple testers access one ECU

Scenario: an [AtomicSwComponentType](#) provides a server port to get information on the status of the protocol communication. Further on the [AtomicSwComponentType](#) may disallow a protocol.

[TPS_SWCT_02016] [AtomicSwComponentType](#) requires information on the status of the protocol communication and may disallow a protocol [

ServiceNeeds kind [DiagnosticCommunicationManagerNeeds](#)

RoleBasedPortAssignment valid roles:

- [CallbackDCMRequestServices](#) [1]

RoleBasedDataAssignment

N/A

RepresentedPortGroups

N/A

]([RS_SWCT_00170](#), [RS_SWCT_03190](#))

For more information please refer to [SWS_Dcm_00692]

7.11.3.7.3.10 Dcm Service Use Case: Service Request Notification

Scenario: an `AtomicSwComponentType` provides a server port to get notified about a Service Request via diagnostic services. This indicates the successful reception of a new request to application.

Within this Service Request Notification this function application can examine the permission of the diagnostic service / environment.

Please note that the Service Request Notification can be used in two characteristics, i.e. as *manufacturer* ([TPS_SWCT_01577] applies) or as a *supplier* ([TPS_SWCT_01578] applies).

[TPS_SWCT_01577] `AtomicSwComponentType` requires the notification about a Service Request via diagnostic services with *manufacturer* characteristics [

The attribute `DiagnosticCommunicationManagerNeeds.serviceRequest-CallbackType` shall be set to the value `requestCallbackTypeManufacturer`.

ServiceNeeds kind `DiagnosticCommunicationManagerNeeds`

RoleBasedPortAssignment valid roles:

- `ServiceRequestNotification` [1]

RoleBasedDataAssignment

N/A

RepresentedPortGroups

N/A

]([RS_SWCT_03190](#))

[TPS_SWCT_01578] `AtomicSwComponentType` requires the notification about a Service Request via diagnostic services with *supplier* characteristics [

The attribute `DiagnosticCommunicationManagerNeeds.serviceRequest-CallbackType` shall be set to the value `requestCallbackTypeSupplier`.

ServiceNeeds kind `DiagnosticCommunicationManagerNeeds`

RoleBasedPortAssignment valid roles:

- `ServiceRequestNotification` [1]

RoleBasedDataAssignment

N/A

RepresentedPortGroups

N/A

|(RS_SWCT_03190)

For more information please refer to [SWS_Dcm_00694]

7.11.3.7.4 OBD related Needs

The `ObdRatioServiceNeeds` describes further properties of the implementation of the Rate Based Monitoring (e.g. `connectionType`) as well as the logical dependencies relevant for the ECU configuration (e.g. `iumpGroup`)

Class	ObdRatioServiceNeeds			
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
Note	Specifies the abstract needs of a component or module on the configuration of OBD Services in relation to a particular "ratio monitoring" which is supported by this component or module.			
Base	ARObject, DiagnosticCapabilityElement , Identifiable , Multilanguage Referrable, Referrable , ServiceNeeds			
Attribute	Datatype	Mul.	Kind	Note
connectionType	ObdRatioConnectionKindEnum	1	attr	Defines how the DEM is connected to the component or module to perform the IUMPR (In use monitor performance ratio) service.
iumpGroup	NameToken	1	attr	Defines the IUMPR (In use monitor performance ratio) Group of the SAE standard. Note that possible values are not predefined by an enumeration meta-type in order to make the meta-model independent of the details of the SAE standard.
rateBasedMonitoredEvent	DiagnosticEventNeeds	1	ref	The rate based monitored Diagnostic Event.
usedFid	FunctionInhibitionNeeds	0..1	ref	This represents the primary Function Inhibition Identifier used for the rate based monitor. This is an optional attribute.
usedSecondaryFid	FunctionInhibitionNeeds	*	ref	This represents the secondary Function Inhibition Identifier used for the rate based monitor. This is an optional attribute. Any of the FID inhibitions leads to an inhibition of the IUMPR calculation

Table 7.94: ObdRatioServiceNeeds

The possible values for the attribute `ObdRatioServiceNeeds.iumpGroup` are:

- CAT1
- CAT2
- OXS1
- OXS2
- EGR

- SAIR
- EVAP
- SECOXS1
- SECOXS2
- NMHCCAT
- NOXCAT
- NOXADSORB
- PMFILTER
- EGSENSOR
- BOOSTPRS
- NOGROUP
- NONE

Enumeration	ObdRatioConnectionKindEnum
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds
Note	Defines the way how the IUMPR service connection between the DEM and the client component or module is handled (for details see the DEM Specification).
Literal	Description
apiUse	The IUMPR service (of the DEM) uses an explicit API to connect to the component or module.
observer	The IUMPR service (of the DEM) uses no API but "observes" the associated diagnostic event.

Table 7.95: ObdRatioConnectionKindEnum

In addition, [ObdPidServiceNeeds](#), [ObdInfoServiceNeeds](#), [ObdMonitorServiceNeeds](#) and [ObdControlServiceNeeds](#) are required in order to specify the specific needs for OBD diagnostic service calls. Note that [ObdPidServiceNeeds](#) is used for the Diagnostic Event Manager as well.

Class	ObdControlServiceNeeds			
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
Note	Specifies the abstract needs of a component or module on the configuration of OBD Service 08 (request control of on-board system) in relation to a particular test-Identifier (TID) supported by this component or module.			
Base	ARObject, DiagnosticCapabilityElement , Identifiable , Multilanguage Referrable, Referrable , ServiceNeeds			
Attribute	Datatype	Mul.	Kind	Note
testId	PositiveInteger	1	attr	Test Identifier (TID) according to ISO 15031-5.

Table 7.96: ObdControlServiceNeeds

Class	ObdPidServiceNeeds			
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
Note	Specifies the abstract needs of a component or module on the configuration of OBD Services in relation to a particular PID (parameter identifier) which is supported by this component or module. In case of using a client/server communicated value, the related value shall be communicated via the port referenced by assignedPort. The details of this communication (e.g. appropriate naming conventions) are specified in the related software specifications (SWS).			
Base	ARObject, DiagnosticCapabilityElement , Identifiable , Multilanguage Referrable, Referrable , ServiceNeeds			
Attribute	Datatype	Mul.	Kind	Note
dataLength	PositiveInteger	0..1	attr	This attribute is applicable only if the ServiceNeeds is aggregated within BswModuleDependency. This attribute represents the length of data (in bytes) provided for this particular PID signal.
parameterId	PositiveInteger	1	attr	Standardized parameter identifier (PID) according to the OBD standard specified in attribute "standard".
standard	String	1	attr	Annotates the standard according to which the PID is given, e.g. "ISO15031-5" or "SAE J1979 Rev May 2007".

Table 7.97: ObdPidServiceNeeds

Class	ObdInfoServiceNeeds			
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
Note	Specifies the abstract needs of a component or module on the configuration of OBD Services in relation to a given InfoType (OBD Service 09) which is supported by this component or module.			
Base	ARObject, DiagnosticCapabilityElement , Identifiable , Multilanguage Referrable, Referrable , ServiceNeeds			
Attribute	Datatype	Mul.	Kind	Note
dataLength	PositiveInteger	0..1	attr	This attribute is applicable only if the ServiceNeeds is aggregated within BswModuleDependency. This attribute represents the length of data (in bytes) provided for this InfoType.
infoType	PositiveInteger	1	attr	The InfoType according to ISO 15031-5

Table 7.98: ObdInfoServiceNeeds

Class	ObdMonitorServiceNeeds			
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
Note	Specifies the abstract needs of a component or module on the configuration of OBD Services in relation to a particular on-board monitoring test supported by this component or module. (OBD Service 06).			
Base	ARObject, DiagnosticCapabilityElement , Identifiable , Multilanguage Referrable, Referrable , ServiceNeeds			
Attribute	Datatype	Mul.	Kind	Note
onBoardMonitorId	PositiveInteger	1	attr	On-board monitor ID according to ISO 15031-5.
testId	PositiveInteger	1	attr	Test Identifier (TID) according to ISO 15031-5.
unitAndScalingId	PositiveInteger	1	attr	Unit and scaling ID according to ISO 15031-5.

Table 7.99: ObdMonitorServiceNeeds

7.11.3.7.4.1 Dem Service Use Case: In-Use-Monitor Performance Ratio calculation

Scenario: an [AtomicSwComponentType](#) implements a OBD system monitor with In-Use-Monitor Performance Ratio (IUMPR) and offers client ports to provide the capability to define the number of times a fault could have been found.

[TPS_SWCT_02007] [AtomicSwComponentType](#) implements a OBD system monitor with In-Use-Monitor Performance Ratio [

ServiceNeeds kind [ObdRatioServiceNeeds](#)

RoleBasedPortAssignment valid roles:

- IUMPRNumerator [0..1]
- IUMPRDenominator [0..1]

RoleBasedDataAssignment

N/A

RepresentedPortGroups

N/A

]([RS_SWCT_00170](#), [RS_SWCT_03190](#))

For more information please refer to [SWS_Dem_00610] and [SWS_Dem_00611].

[constr_2053] Consistency between [role](#) IUMPRNumerator and [ObdRatioServiceNeeds.connectionType](#) [If a [SwcServiceDependency](#) with a [ObdRatioServiceNeeds](#) is defined and the attribute [connectionType](#) of the contained [ObdRatioServiceNeeds](#) is set to [ObdRatioConnectionKindEnum.apiUse](#) a [RoleBasedPortAssignment](#) with the [role](#) value IUMPRNumerator shall be defined.

If the attribute `connectionType` of the contained `ObdRatioServiceNeeds` is set to `ObdRatioConnectionKindEnum.observer` the `role` value `IUMPRNumerator` is not applicable. `]()`

7.11.3.7.4.2 Dcm Service Use Case: read parameter identifier via diagnostic services by Client Server Interface

Scenario: an `AtomicSwComponentType` offers a server port to read/write current value via OBD services.

[TPS_SWCT_02008] `AtomicSwComponentType` offers a server port to read/write current value via OBD services `[`

ServiceNeeds kind `ObdPidServiceNeeds`

RoleBasedPortAssignment

The following roles are applicable:

- `DataServices` [1]

RoleBasedDataAssignment

N/A

RepresentedPortGroups

N/A

`](RS_SWCT_00170, RS_SWCT_03190)`

[TPS_SWCT_01630] Suffix used for the resulting name of the `PortInterface` for the Data Services `[` The *suffix* used for the resulting name of the `PortInterface` for the Data Services (`DataServices_{Data}`) shall be taken from the `shortName` of the applicable `SwcServiceDependency`. `](RS_SWCT_00170, RS_SWCT_03190)`

For more information please refer to [SWS_Dcm_00686].

7.11.3.7.4.3 Dcm Service Use Case: read parameter identifier via diagnostic services by Sender Receiver Interface

Scenario: an `AtomicSwComponentType` offers sender receiver ports to read/write current values via OBD services.

[TPS_SWCT_02009] `AtomicSwComponentType` offers sender receiver ports to read/write current values via OBD services `[`

ServiceNeeds kind `ObdPidServiceNeeds`

RoleBasedPortAssignment

N/A

RoleBasedDataAssignment

The following roles are applicable:

- `signalBasedDiagnostics` [1..2]

RepresentedPortGroups

N/A

To read the signal the `AtomicSwComponentType` shall offer an `AbstractProvidedPortPrototype`, to write the signal the `AtomicSwComponentType` shall offer an `AbstractRequiredPortPrototype`. [\]\(RS_SWCT_00170, RS_SWCT_03190\)](#)

For more information please refer to [SWS_Dcm_00687].

7.11.3.7.4.4 Dcm Service Use Case: Request vehicle information

Scenario: an `AtomicSwComponentType` offers a server port to read vehicle information values via OBD services.

[TPS_SWCT_02010] `AtomicSwComponentType` offers a server port to read vehicle information values via OBD services [\[](#)

ServiceNeeds kind `ObdInfoServiceNeeds`

RoleBasedPortAssignment valid roles:

- `InfotypeServices` [1]

RoleBasedDataAssignment

N/A

RepresentedPortGroups

N/A

[\]\(RS_SWCT_00170, RS_SWCT_03190\)](#)

[TPS_SWCT_01631] **Suffix used for the resulting name of the `PortInterface` for the Infotype Services** [\[](#) The *suffix* used for the resulting name of the `PortInterface` for the Infotype Services (`InfotypeServices_{VehInfoData}`) shall be taken from the *shortName* of the applicable `SwcServiceDependency`. [\]\(RS_SWCT_00170, RS_SWCT_03190\)](#)

For more information please refer to [SWS_Dcm_00688].

7.11.3.7.4.5 Dem Service Use Case: Read DTR data from SW-C for OBD Service \$06

Scenario: an `AtomicSwComponentType` offers a server ports to read DTR value via OBD services.

[TPS_SWCT_02011] [AtomicSwComponentType](#) offers a server port to read DTR value via OBD services [

ServiceNeeds kind [ObdMonitorServiceNeeds](#)

RoleBasedPortAssignment valid roles:

- [DTRCentralReport](#) [1]

RoleBasedDataAssignment

N/A

RepresentedPortGroups

N/A

]([RS_SWCT_00170](#), [RS_SWCT_03190](#))

For more information please refer to [SWS_Dcm_00689].

7.11.3.7.4.6 Dcm Service Use Case: request control of on-board system, test or component

Scenario: an [AtomicSwComponentType](#) offers a server port for request control of on-board system, test or component via OBD services.

[TPS_SWCT_02012] [AtomicSwComponentType](#) offers a server port for request control of on-board system, test or component via OBD services [

ServiceNeeds kind [ObdControlServiceNeeds](#)

RoleBasedPortAssignment

The following roles are applicable:

- [RequestControlServices](#) [1]

RoleBasedDataAssignment

N/A

RepresentedPortGroups

N/A

]([RS_SWCT_00170](#), [RS_SWCT_03190](#))

[TPS_SWCT_01633] **Suffix used for the resulting name of the [PortInterface](#) for the Request Control Services** [The *suffix* used for the resulting name of the [PortInterface](#) for the Request Control Services ([RequestControlServices_{Tid}](#)) shall be taken from the [shortName](#) of the applicable [SwcServiceDependency](#).]([RS_SWCT_00170](#), [RS_SWCT_03190](#))

For more information please refer to [SWS_Dcm_00691].

7.11.3.7.4.7 Dcm Service Use Case: Response On Event via diagnostic services

Please note that for this specific use case the application of a concrete [ServiceNeeds](#) is not yet clarified.

Scenario: an [AtomicSwComponentType](#) offers client server ports to support Response On Event (ROE) via diagnostic services.

[TPS_SWCT_02014] [AtomicSwComponentType](#) supports Response On Event (ROE) via diagnostic services [

RoleBasedPortAssignment valid roles:

- Dcm_Roe [1]

RoleBasedDataAssignment

N/A

RepresentedPortGroups

N/A

The role `ROEServices` is applicable for a server port of the [AtomicSwComponentType](#) and the role `Dcm_Roe` is applicable for a client port of the [AtomicSwComponentTypes](#)]([RS_SWCT_00170](#), [RS_SWCT_03190](#))

For more information please refer to [SWS_Dcm_00695] and [SWS_Dcm_00699].

7.11.3.7.5 Diagnostics over IP

This chapter describes the usage of specific meta-classes to support the specification of diagnostics over IP. For more details, please refer to ISO 13400 [39].

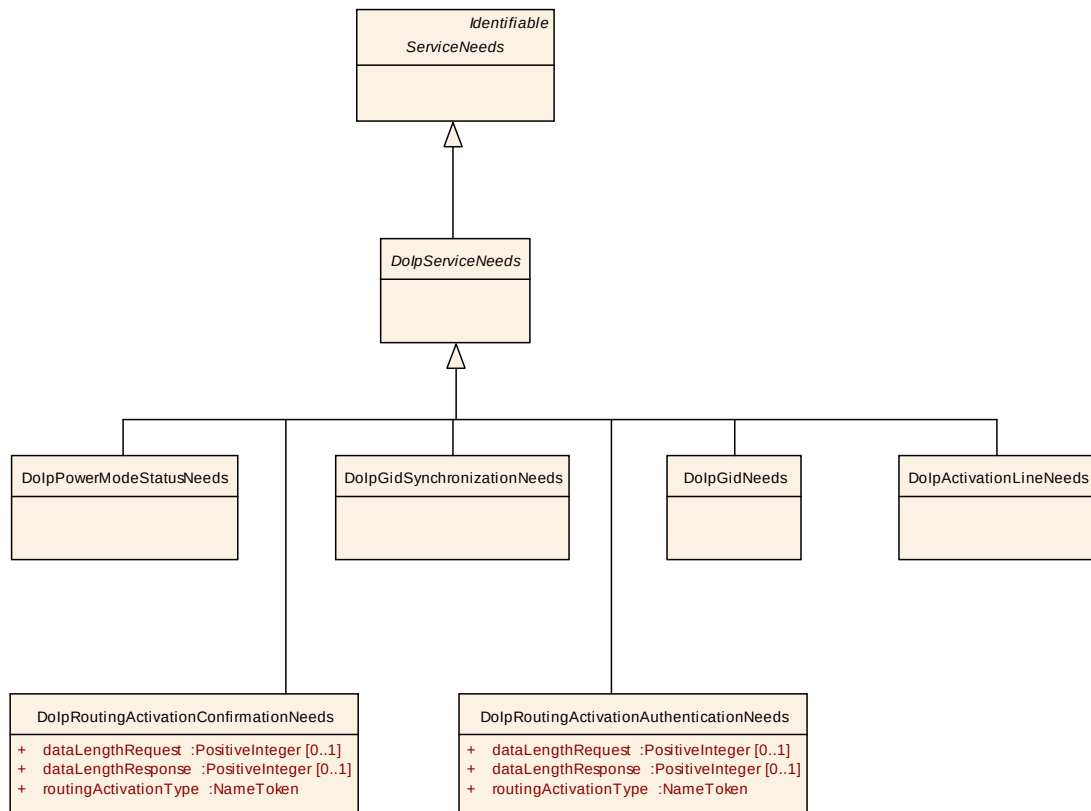


Figure 7.47: Subclasses of **ServiceNeeds** for implementing diagnostics over IP

Class	DolpServiceNeeds (abstract)			
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
Note	This represents an abstract base class for ServiceNeeds related to DoIP.			
Base	ARObject, Identifiable , MultilanguageReferrable, Referrable , ServiceNeeds			
Attribute	Datatype	Mul.	Kind	Note
—	—	—	—	—

Table 7.100: DolpServiceNeeds

Class	DolpGidNeeds			
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
Note	The DolpGidNeeds indicates that the software-component owning this ServiceNeeds is providing the GID number either after a GID Synchronisation or by other means like e.g. flashed EEPROM parameter. This need can be used independent from DolpGidSynchronizationNeeds and is necessary if the GID can not be provided out of the DoIP configuration options.			
Base	ARObject, DolpServiceNeeds , Identifiable , MultilanguageReferrable, Referrable , ServiceNeeds			
Attribute	Datatype	Mul.	Kind	Note
—	—	—	—	—

Table 7.101: DolpGidNeeds

Class	DolpGidSynchronizationNeeds			
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
Note	The DolpGidSynchronizationNeeds indicates that the software-component owning this ServiceNeeds is triggered by the DoIP entity to start a synchronization of the GID (Group Identification) on the DoIP service 0x0001, 0x0002, 0x0003 or before announcement via service 0x0004 according to ISO 13400-2:2012 if necessary. Note that this need is only relevant for DoIP synchronization masters.			
Base	ARObject, DolpServiceNeeds , Identifiable , MultilanguageReferrable, Referrable , ServiceNeeds			
Attribute	Datatype	Mul.	Kind	Note
–	–	–	–	–

Table 7.102: DolpGidSynchronizationNeeds

Class	DolpPowerModeStatusNeeds			
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
Note	The DolpPowerModeStatusNeeds indicates that the software-component owning this ServiceNeeds is providing the PowerModeStatus for the DoIP service 0x4003 according to ISO 13400-2:2012.			
Base	ARObject, DolpServiceNeeds , Identifiable , MultilanguageReferrable, Referrable , ServiceNeeds			
Attribute	Datatype	Mul.	Kind	Note
–	–	–	–	–

Table 7.103: DolpPowerModeStatusNeeds

Class	DolpRoutingActivationAuthenticationNeeds			
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
Note	DoIPRoutingActivationAuthenticationNeeds indicates that the software-component owning this ServiceNeeds will have an authentication required for a DoIP routing activation service (0x0005) according to ISO 13400-2:2012.			
Base	ARObject, DolpServiceNeeds , Identifiable , MultilanguageReferrable, Referrable , ServiceNeeds			
Attribute	Datatype	Mul.	Kind	Note
dataLength Request	PositiveInteger	0..1	attr	Describes the length in byte of the additional information for RA authentication that is needed by the software entity. If the software entity is a SWC the attribute does not need to exist as the information is available via the length of the uint8 Array type. Otherwise (i.e the software entity is a Complex Driver) this attribute needs to be filled out if additional information is needed.
dataLength Response	PositiveInteger	0..1	attr	Describes the length in byte of the additional information for RA authentication that is provided by the software entity. If the software entity is a SWC the attribute does not need to exist as the information is available via the length of the uint8 Array type. Otherwise (i.e the software entity is a Complex Driver) this attribute needs to be filled in if additional information is provided.

Attribute	Datatype	Mul.	Kind	Note
routingActivationType	NameToken	1	attr	Describes the ISO 13400-2:2012 "routing activation request activation type" which is received via DoIP service 0x0005. 0x00 is DEFAULT, 0x01 is WWH-OBD. If neither of the specified values (0x00 or 0x01) is needed the token shall contain RA_ + hex value representation of the integer value shall be used (i.e: RA_0xE1).

Table 7.104: DoIPRoutingActivationAuthenticationNeeds

Class	DoIPRoutingActivationConfirmationNeeds			
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
Note	DoIPRoutingActivationConfirmationNeeds indicates that the software-component that owns this ServiceNeeds will have a confirmation required for a DoIP routing activation service (0x0005) according to ISO 13400-2:2012.			
Base	ARObject, DoIPServiceNeeds , Identifiable , MultilanguageReferrable , Referrable , ServiceNeeds			
Attribute	Datatype	Mul.	Kind	Note
dataLengthRequest	PositiveInteger	0..1	attr	Describes the length in byte of the additional information for RA confirmation that is needed by the software entity. If the software entity is a SWC the attribute does not need to exist as the information is available via the length of the uint8 Array type. Otherwise (i.e the software entity is a Complex Driver) this attribute needs to be filled out if additional information is needed.
dataLengthResponse	PositiveInteger	0..1	attr	Describes the length in byte of the additional information for RA confirmation that is provided by the software entity. If the software entity is a SWC the attribute does not need to exist as the information is available via the length of the uint8 Array type. Otherwise (i.e the software entity is a Complex Driver) this attribute needs to be filled out if additional information is provided.
routingActivationType	NameToken	1	attr	Describes the ISO 13400-2:2012 "routing activation request activation type" which is received via DoIP service 0x0005. 0x00 is DEFAULT, 0x01 is WWH-OBD. If neither of the specified values (0x00 or 0x01) is needed the token shall contain RA_ + hex value representation of the integer value shall be used (i.e: RA_0xE1).

Table 7.105: DoIPRoutingActivationConfirmationNeeds

Class	DolpActivationLineNeeds			
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
Note	A DoIP entity needs to be informed when an external tester is attached or activated. The DolpActivationServiceNeeds specifies the trigger for such an event. Examples would be a Pdu via a regular communication bus, a PWM signal, or an I/O. For details please refer to the ISO 13400.			
Base	ARObject, DolpServiceNeeds , Identifiable , MultilanguageReferrable, Referrable , ServiceNeeds			
Attribute	Datatype	Mul.	Kind	Note
–	–	–	–	–

Table 7.106: DolpActivationLineNeeds

7.11.3.7.5.1 DoIP Service Use Case: GID synchronization can be necessary if the ECU is DoIP Gid synchronization master

Scenario: on the event of connecting a tester to an ECU a GID synchronization can be necessary if the ECU is DoIP Gid synchronization master. In this case, it is necessary to define a [DoIpGidSynchronizationNeeds](#).

[TPS_SWCT_01537] GID synchronization can be necessary if the ECU is DoIP Gid synchronization master [

ServiceNeeds kind [DoIpGidSynchronizationNeeds](#)

RoleBasedPortAssignment valid roles:

- [CallbackTriggerGIDSynchronization](#) [1]

RoleBasedDataAssignment

N/A

RepresentedPortGroups

N/A

] ([RS_SWCT_03310](#), [RS_SWCT_03190](#))

7.11.3.7.5.2 DoIP Service Use Case: Vehicle information is broadcast or can be requested by the tester

Scenario: vehicle information is broadcast or can be requested by the tester. In this case, it is necessary to define a [DoIpGidNeeds](#).

[TPS_SWCT_01538] Vehicle information is broadcast or can be requested by the tester [

ServiceNeeds kind [DoIpGidNeeds](#)

RoleBasedPortAssignment valid roles:

- CallbackGetGID [1]

RoleBasedDataAssignment

N/A

RepresentedPortGroups

N/A

]([RS_SWCT_03310](#), [RS_SWCT_03190](#))

7.11.3.7.5.3 DoIP Service Use Case: Tester could also request the power status with respect to diagnostics

Scenario: before starting the diagnostics processing for the DoIP entity or sub-networks connected via DoIP, the tester could also request the power status with respect to diagnostics. To support this option it will be necessary to define a [DoIpPowerModeStatusNeeds](#).

[TPS_SWCT_01539] Tester can also request before starting diagnostic processing for the DoIP entity or sub-networks connected via DoIP the power status with respect to diagnostics [

ServiceNeeds kind [DoIpPowerModeStatusNeeds](#)

RoleBasedPortAssignment valid roles:

- CallbackGetPowerMode [1]

RoleBasedDataAssignment

N/A

RepresentedPortGroups

N/A

]([RS_SWCT_03310](#), [RS_SWCT_03190](#))

7.11.3.7.5.4 DoIP Service Use Case: Routing activation mechanism is used which can lead to additional impact regarding authentication or confirmation

Scenario: to enable diagnostics of the tester to a different target address, the routing activation mechanism is used which can lead to additional impact regarding authentication or confirmation. Here, the definition of [DoIpRoutingActivationAuthenticationNeeds](#) and/or [DoIpRoutingActivationConfirmationNeeds](#) would be applicable.

[TPS_SWCT_01544] *prefix* used for the actual name of the used [PortInterface](#) for the routing activation [The *prefix* used for the actual name of the used [PortIn-](#)

terface for the routing activation shall be taken from the `shortName` of the enclosing `SwcServiceDependency`. `](RS_SWCT_03310, RS_SWCT_03190)`

[TPS_SWCT_01540] Routing activation mechanism is used which can lead to additional impact regarding authentication or confirmation `[`

ServiceNeeds kind

- `DoIpRoutingActivationAuthenticationNeeds` `[0..1]`
- `DoIpRoutingActivationConfirmationNeeds` `[0..1]`

RoleBasedPortAssignment valid roles:

- `RoutingActivation` `[1]`

RoleBasedDataAssignment

N/A

RepresentedPortGroups

N/A

`](RS_SWCT_03190)`

7.11.3.7.5.5 DoIP Service Use Case: a DoIP entity needs to be informed when an external tester is attached or activated.

Scenario: to enable diagnostics by connecting a tester to an ECU it is necessary that the application software becomes aware of the tester's presence.

For this purpose, the applicable `ServiceSwComponentType` is supposed to provide a `PPortPrototype` typed by the `ModeSwitchInterface` named `DoIPActivationLineStatus` towards the application.

To trigger the existence of the `PPortPrototype`, `DoIpActivationLineNeeds` shall be defined.

[TPS_SWCT_01546] Notification when an external tester is attached or activated `[`

ServiceNeeds kind `DoIpActivationLineNeeds`

RoleBasedPortAssignment valid roles:

- `DoIPActivationLineStatus` `[1]`

RoleBasedDataAssignment

N/A

RepresentedPortGroups

N/A

]([RS_SWCT_03310](#), [RS_SWCT_03190](#))

7.11.3.7.5.6 Service Use Case: Set and reset Warning Indicator Request bit

Scenario: In some cases (e.g. controlling a failsafe reaction in application) the “Warning Indicator Request”-bit of a corresponding event in Dem shall be set/reset by a special “failsafe software-component”.

The failsafe software-component has to ensure a proper status of the “Warning Indicator Request”-bit (e.g. regarding to ISO14229-1 or manufacture specific requirements).

Therefore the failsafe SW-C can use existing Dem mechanism to get the information about status changes of events in Dem (e.g. Callback `EventStatusChanged`).

For this purpose, the applicable `ServiceSwComponentType` is supposed to provide a `PPortPrototype` typed by the `ClientServerInterface` named `EventStatus` towards the application.

To trigger the existence of the `PPortPrototype`, `WarningIndicatorRequestedBitNeeds` shall be defined.

[[TPS_SWCT_01547](#)] Ability to set and reset the Warning Indicator Request bit [

ServiceNeeds kind `WarningIndicatorRequestedBitNeeds`

RoleBasedPortAssignment valid roles:

- `EventStatus` [1]

RoleBasedDataAssignment

N/A

RepresentedPortGroups

N/A

]([RS_SWCT_03190](#))

Class	WarningIndicatorRequestedBitNeeds			
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
Note	This meta-class represents the ability to explicitly request the existence of the <code>WarningIndicatorRequestedBit</code> .			
Base	ARObject, DiagnosticCapabilityElement , Identifiable , Multilanguage Referrable, Referrable , ServiceNeeds			
Attribute	Datatype	Mul.	Kind	Note
—	—	—	—	—

Table 7.107: `WarningIndicatorRequestedBitNeeds`

7.11.3.8 Diagnostic Log and Trace Dependency

The meta-class [DltUserNeeds](#) is used together with the [SwcServiceDependency](#) to define requirements in order to configure the Diagnostic Log and Trace module (for the terms related to the AUTOSAR Specification of Module DLT see [40]).

Class	DltUserNeeds			
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
Note	Specifies the needs on the configuration of the Diagnostic Log and Trace module for one SessionId. This class currently contains no attributes. An instance of this class is used to find out which ports of an SWC belong to this SessionId in order to group the request and response ports of the same SessionId. The actual SessionId value is stored in the PortDefinedArgumentValue of the respective port specification.			
Base	ARObject, Identifiable , MultilanguageReferrable, Referrable , ServiceNeeds			
Attribute	Datatype	Mul.	Kind	Note
–	–	–	–	–

Table 7.108: DltUserNeeds

Please note that for the described use case of the Dlt Service the following rule applies: For every used [ClientServerInterface](#) it is necessary to create a [RoleBasedPortAssignment](#).

Thereby the value of the attribute [role](#) of the [RoleBasedPortAssignment](#) has to be set to the name of the used standardized [ClientServerInterface](#).

The possible role attribute values and the multiplicity of the related [PortPrototypes](#) are listed at the use case descriptions in the paragraph **RoleBasedPortAssignment**.

7.11.3.8.1 Dlt use Case: Application software component accesses the Synchronized Time-Base Manager

Scenario: [AtomicSwComponentType](#) sends log messages. In this case the following setup applies:

[TPS_SWCT_02506] Setup for Dlt use Case: Application software component accesses the Synchronized Time-Base Manager [

RoleBasedPortAssignment valid roles:

- [DLTService](#) [1]
- [LogTraceSessionControl](#) [1]
- [VerboseModeControl](#) [0..1]
- [InjectionCallback](#) [0..1]

RoleBasedDataAssignment

N/A

RepresentedPortGroups

N/A

]()

For more information please refer to [SWS_Dlt_00495], [SWS_Dlt_00496], [SWS_Dlt_00497], and [SWS_Dlt_00498].

In this case the software-component has to provide one Client Port (DLTService) in order to register the context and to send log or trace messages.

Further on the component has to provide a Server Port (LogTraceSessionControl) to receive the current log level and trace status. Server Ports for `VerboseModeControl` and `InjectionCallback` are optional.

7.11.3.9 Synchronized Time-Base Manager Dependency

The meta-class `SyncTimeBaseMgrUserNeeds` is used together with the `SwcServiceDependency` to define requirements in order to configure the Synchronized Time-Base Manager module (for the terms related to the AUTOSAR Specification of Module StbM see [41]).

Class	SyncTimeBaseMgrUserNeeds			
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
Note	Specifies the needs on the configuration of the Synchronized Time-base Manager for one time-base. This class currently contains no attributes. An instance of this class is used to find out which ports of a software-component belong to this time-base in order to group the request and response ports of the same time-base. The actual time-base value is stored in the PortDefinedArgumentValue of the respective port specification.			
Base	ARObject, Identifiable , MultilanguageReferrable, Referrable , ServiceNeeds			
Attribute	Datatype	Mul.	Kind	Note
—	—	—	—	—

Table 7.109: SyncTimeBaseMgrUserNeeds

Please note that for the described use cases of the StbM Service following rule applies: For every used `ClientServerInterface` it is necessary to create a `RoleBasedPortAssignment`.

Thereby the value of the attribute `role` of the `RoleBasedPortAssignment` has to be set to the name of the used standardized `ClientServerInterface`.

The possible role attribute values and the multiplicity of the related `PortPrototypes` are listed at the use case descriptions in the paragraph **RoleBasedPortAssignment**.

7.11.3.9.1 StbM use Case: Application software component accesses the Synchronized Time-Base Manager

Scenario: an `AtomicSwComponentType` autonomously calls the Synchronized Time-Base Manager, getting knowledge about the definition of time and the state of the module. In this case the following setup applies:

RoleBasedPortAssignment valid roles:

- `StbM_TimeBaseValue` [1]

RoleBasedDataAssignment

N/A

RepresentedPortGroups

N/A

In this case the software component has to provide one Client Port (`StbM_TimeBaseValue`) to

- access the current status of the synchronized time-base
- access the current definition of time represented by the notion of ticks
- access the current definition of tickDuration

7.11.3.9.2 StbM use Case: Synchronized Time-Base Manager notifies application software component

Scenario: an `AtomicSwComponentType` shall be informed by the Synchronized Time-Base Manager about state changes and/or error occurrences (e.g. the synchronisation state of a time-base has changed). In this case the following setup applies:

RoleBasedPortAssignment valid roles:

- `StbM_TimeBase_TriggerCustomer` [0..1]
- `StbM_TimeBase_StateNotification` [0..1]

RoleBasedDataAssignment

N/A

RepresentedPortGroups

N/A

In this case the software-component has to provide one Receiver Port (`StbM_TimeBase_TriggerCustomer`) to receive the current value of the synchronized time-base and / or one Receiver Port (`StbM_TimeBase_StateNotification`) to receive the current state of the synchronized time-base.

Please note that at least one of the two possible `RoleBasedPortAssignments` shall exist for this use case.

7.12 Variation Point Proxy

[TPS_SWCT_01370] **VariationPointProxy** [Variability inside a software-component may exist in two different levels of abstraction:

- A **structural** variation point affects the existence or non-existence of structural model elements. A structural variation point is modeled by means of the meta-class **VariationPoint**.
- A **functional** variation point affects solely the functionality in the implementation (read: source code) of the software-component. A functional variation point is modeled by means of the meta-class **VariationPointProxy**.

In other words, this enables the developer of a software-component to implement variability that is limited to the software-component's functionality. This kind of variability is resolved

- by a code generator (`bindingTime = codeGenerationTime`)
- by the preprocessor (`bindingTime = preCompileTime`).
- as a post-build value evaluation (in this case `postBuildValueAccess` and `postBuildVariantCondition` shall exist).

]([RS_SWCT_03100](#))

Please note that in the first two cases of the second bullet list in [TPS_SWCT_01370] the evaluation of `conditionAccess` shall replace the formula by the result.

The name **VariationPointProxy** was motivated by the fact that it represents a model element that is not directly related to the **structure** but to the code and from this point of view acts as a proxy to the **functional** variation existing in the code.

The consequence of the two levels of abstraction is that (from a model processing point of view) it would be possible to bind all structural variation points entirely while keeping some or all of the functional variation points unbound. This is an explanation for the existence of [TPS_SWCT_01371].

[TPS_SWCT_01371] **VariationPointProxy vs. VariationPoint** [The difference between a **VariationPoint** and a **VariationPointProxy** is that if during the process of binding the formula evaluates to 0 the **VariationPointProxy** remains in the model while the **VariationPoint** as well as its owner is removed from the model.]([RS_SWCT_03100](#))

Nevertheless, the binding of the variability is described by the means of `SwSystem-constantValueSets` and `PostBuildVariantCriterionValueSets`.

[TPS_SWCT_01448] **Pre-defined values for the category of VariationPointProxy** [AUTOSAR pre-defines two possible values for the `category` of **VariationPointProxy**. The meaning of the values, however, depends on the particular modeling of individual **VariationPointProxies**, see [TPS_SWCT_01370].

VALUE In the “pre-build” case this means that `valueAccess` shall yield an integer literal. In the “post-build” case, on the other hand, this means that `postBuildValueAccess` shall yield an integer value conform with the `implementationDataType`.

In this context, `[constr_1388]` applies.

CONDITION In this case it is **possible** (though not mandatory) to define a `VariationPointProxy` that actually works in a combination of the “pre-build” and “post-build” scenario.

In other words, in the “pre-build” case `conditionAccess` shall yield a *boolean* value and in the “post-build” case `postBuildVariantCondition` shall also yield a *boolean* value.

An *and* operator shall be applied to all boolean values returned by `conditionAccess` and the collection of `postBuildVariantCondition` in order to yield the actual result of the condition. `[TPS_GST_00259]` and `[SWS_Rte_08069]` apply.

For the `postBuildVariantCondition` an implicit reference to the `PlatformData Type boolean` shall be assumed.

In contrast to the value `VALUE` it is possible to define a `VariationPointProxy` that uses **both** `conditionAccess` and `postBuildVariantCondition`.

](*RS_SWCT_03100*)

[constr_1388] `VariationPointProxy` of category `VALUE` shall not mix “pre-build” and “post-build” use-cases [If the value of `category` of the `VariationPointProxy` is set to `VALUE` then there can only be one value yield from the evaluation of a `VariationPointProxy`. In other words, a `VariationPointProxy` of `category VALUE` shall not mix the “pre-build” and “post-build” use-cases.]()

[constr_1389] Restriction regarding the value of `category` of `VariationPointProxy.implementationDataType` [`VariationPointProxy.implementationDataType` shall **not** be of `category STRUCTURE`, `ARRAY`, `UNION`, `FUNCTION_REFERENCE`, and `DATA_REFERENCE`. The `VariationPointProxy.implementationDataType` shall be of `category VALUE` or `TYPE_REFERENCE` that, after all references are resolved, yields an `ImplementationDataType` of `category VALUE`.]()

[TPS_SWCT_01372] `bindingTime = preCompileTime` [In case of `bindingTime = preCompileTime` the RTE provides macro definitions that can be used for preprocessor directives to implement `preCompileTime` variability in C/C++ code.](*RS_SWCT_03100*)

[TPS_SWCT_01373] RTE generator shall evaluate the `SwSystemconstDependentFormula` [It is in the scope of the RTE generator to evaluate the `SwSystemconstDependentFormula` which has a higher precedence than the standard C Prepro-

cessor and to provide the resulting values to the software-component's implementation.
](RS_SWCT_03100)

For further details (beyond the statements made in [TPS_SWCT_01372] and [TPS_SWCT_01373]) about the impact of the existence of a [VariationPointProxy](#) on the RTE please refer to [2].

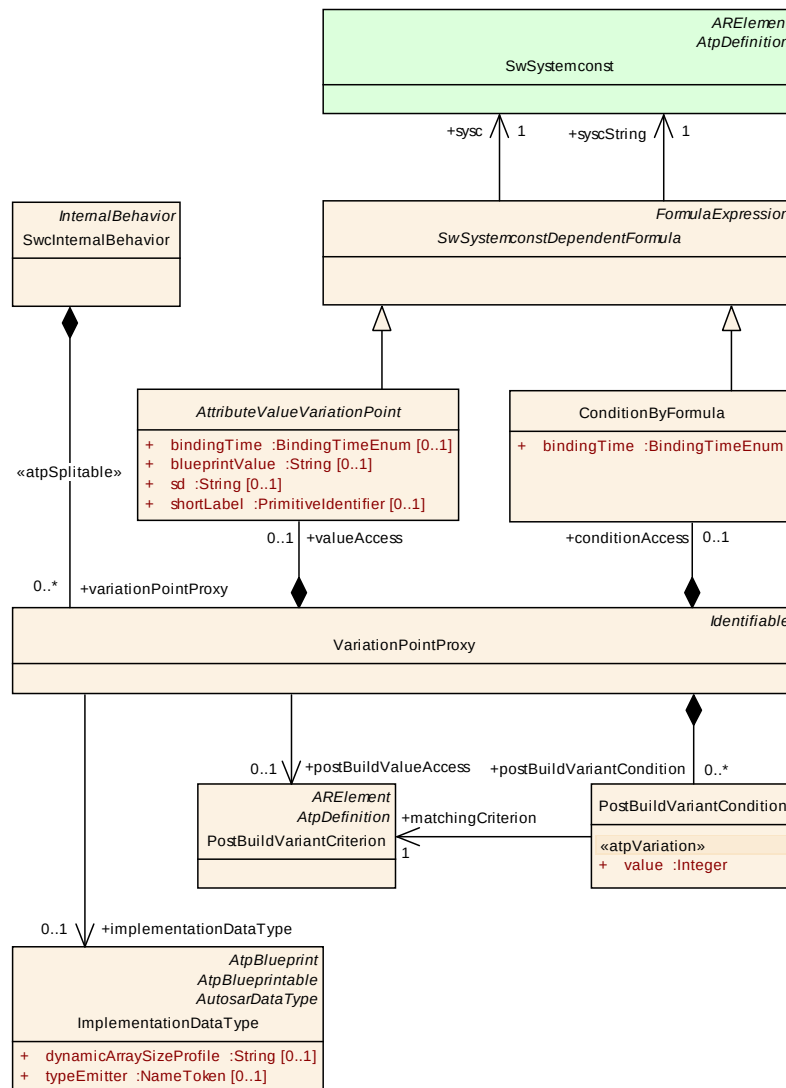


Figure 7.48: VariationPointProxy

Class	VariationPointProxy			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::Variant Handling			
Note	The VariationPointProxy represents variation points of the C/C++ implementation. In case of bindingTime = compileTime the RTE provides defines which can be used for Pre Processor directives to implement compileTime variability.			
Base	ARObject, Identifiable , MultilanguageReferrable, Referrable			
Attribute	Datatype	Mul.	Kind	Note
conditionAccess	ConditionByFormula	0..1	aggr	This condition acts as Binding Function for the VariationPoint.

Attribute	Datatype	Mul.	Kind	Note
implementationDataType	ImplementationDataType	0..1	ref	This association to ImplementationDataType shall be taken as an implementation hint by the RTE generator.
postBuildValueAccesses	PostBuildVariantCriterion	0..1	ref	This represents the applicable PostBuildVariantCriterion in the context of a VariationPointProxy. Note that the technical details how to access the particular postBuildValueAccess are still considered internal to the RTE and are consequently not standardized.
postBuildVariantCondition	PostBuildVariantCondition	*	aggr	This represents that applicable PostBuildVariantCondition in the context of a VariationPointProxy.
valueAccesses	AttributeValueVariationPoint	0..1	aggr	This value acts as Binding Function for the VariationPoint.

Table 7.110: VariationPointProxy

Please note that the usage of attributes of meta-class [VariationPointProxy](#) is not arbitrarily possible but subject to conditions. In particular, there are certain use-cases that dictate how and with which multiplicity attributes of [VariationPointProxy](#) shall be used.

In particular, the applicable use-cases are defined by a combination of the binding time, i.e. *PreBuild* (all pre-build binding times are summarized as *PreBuild*) vs. *PostBuild*, and the value of [VariationPointProxy.category](#) (the details are explained in table 7.111 resp. [constr_1253]).

[constr_1253] Supported usage of [VariationPointProxy](#) [The allowed multiplicities for attributes of [VariationPointProxy](#) depending on the applicable binding time and the value of [VariationPointProxy.category](#) are documented in Table 7.111.

For clarification, the multiplicities of attributes of meta-class [VariationPointProxy](#) that are **not** explicitly mentioned in a given row of table 7.111 shall be interpreted as [0].]()

BindingTime	category	Allowed Attribute Multiplicity
<i>PreBuild</i>	VALUE	valueAccess [1]
	CONDITION	conditionAccess [1]
<i>PostBuild</i>	VALUE	postBuildValueAccess [1], implementationDataType [1]
	CONDITION	postBuildVariantCondition [1..*], conditionAccess [0..1]

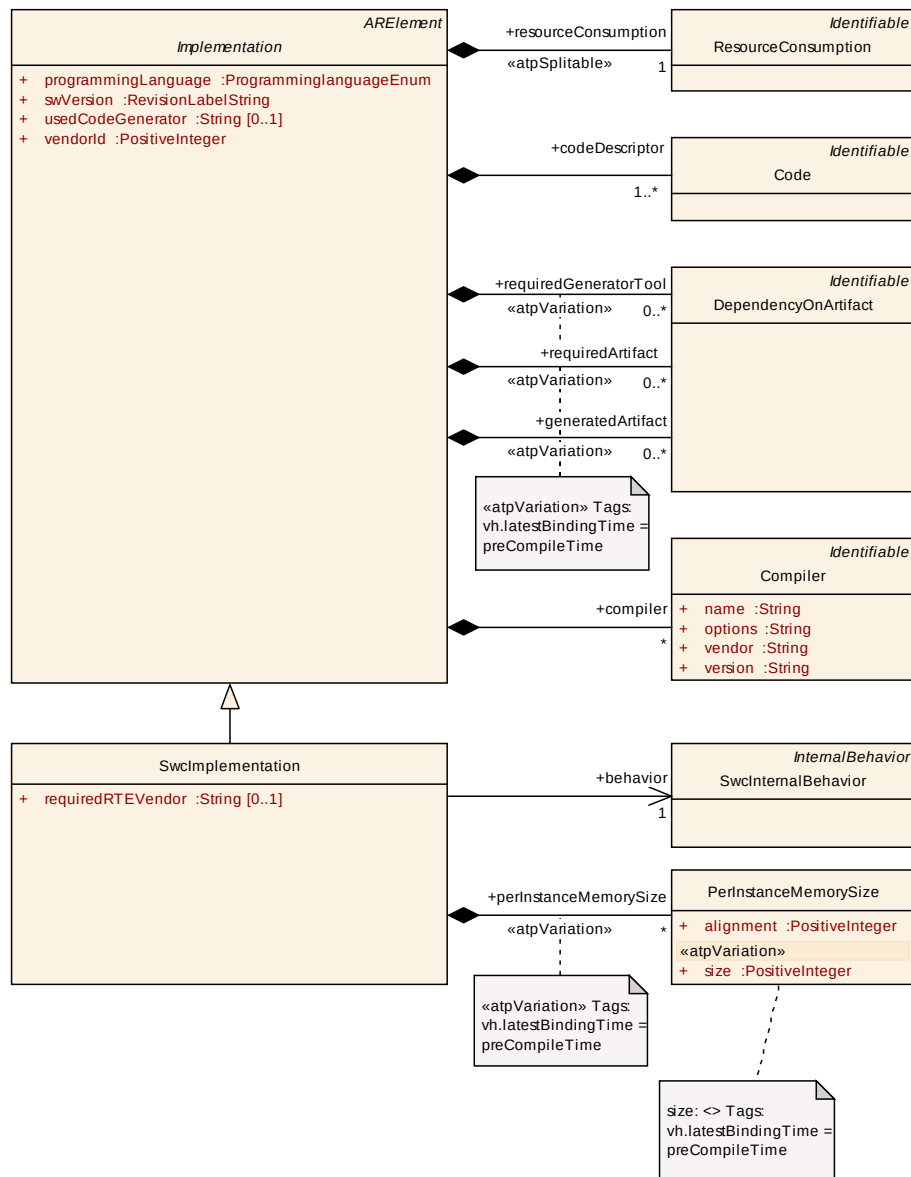
Table 7.111: Supported usage of [VariationPointProxy](#)

8 Implementation

Previous versions of this document contained a comprehensive description of the meta-class `Implementation`. This meta-class still exists but the description of most of its content has been moved to another document, in particular the specification of the Basic Software Module Description Template [7].

Please note that the Software Component Template and the Basic Software Module Description Template share the content of `Implementation`. However, the semantics of `Implementation` is closer to the Basic Software Module Description Template.

Nevertheless, there is still content strictly related to the Software Component Template. This part of `Implementation` consisting of `SwcImplementation` (see Figure 8.1) remains in this document.



Class	SwcImplementation			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcImplementation			
Note	This meta-class represents a specialization of the general Implementation meta-class with respect to the usage in application software. Tags: atp.recommendedPackage=SwcImplementations			
Base	ARElement , ARObject , CollectableElement , Identifiable , Implementation , MultilanguageReferrable , PackageableElement , Referrable			
Attribute	Datatype	Mul.	Kind	Note
behavior	SwcInternalBehavior	1	ref	The internal behavior implemented by this Implementation.
perInstanceMemorySize	PerInstanceMemorySize	*	aggr	Allows a definition of the size of the per-instance memory for this implementation. The aggregation of PerInstanceMemorySize is subject to variability with the purpose to support variability in the software components implementations. Typically different algorithms in the implementation are requiring different number of memory objects, in this case PerInstanceMemory. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime
requiredRTEVendor	String	0..1	attr	Identify a specific RTE vendor. This information is potentially important at the time of integrating (in particular: linking) the application code with the RTE. The semantics is that (if the association exists) the corresponding code has been created to fit to the vendor-mode RTE provided by this specific vendor. Attempting to integrate the code with another RTE generated in vendor mode is in general not possible.

Table 8.1: SwcImplementation

Class	PerInstanceMemorySize			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcImplementation			
Note	Resources needed by the allocation of PerInstanceMemory for each SWC instance. Note that these resources are not covered by an ObjectFileSection, because they are supposed to be allocated by the RTE.			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
alignment	PositiveInteger	1	attr	Required alignment (1,2,4,...) of the referenced PerInstanceMemory. Unit: byte.
perInstanceMemory	PerInstanceMemory	1	ref	This represents the referenced PerInstanceMemory.

Attribute	Datatype	Mul.	Kind	Note
size	PositiveInteger	1	attr	Size (in bytes) of the reference perInstanceMemory. The aggregation of PerInstanceMemorySize is subject to variability with the purpose to support variability in the software components implementations. Different algorithms in the implementation might require a different PerInstanceMemorySize. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime

Table 8.2: PerInstanceMemorySize

9 Mode Management

In general, the Software Component Template doesn't define the kind of modes that shall be supported by State Managers or software-components explicitly. However the Software Component Template provides generic mechanisms for describing modes.

In this section the general relationship between modes, interfaces, and software-components is discussed.

The assumption from the software-component point of view is that State Managers are using a Standardized AUTOSAR [PortInterface](#)¹ to influence the [SwComponentType](#) and also provide a [PortInterface](#) to get requests and confirmations from the [SwComponentType](#).

They will be implemented as AUTOSAR services and be part of the Basic Software on each ECU. The actual modes a State Manager provides will have to be standardized as well to allow compatibility between software-components.

It is also possible to define a mode manager in the Application Software and the same functionality is supported as for mode managers implemented in the Basic Software.

[TPS_SWCT_01581] Communication patterns for mode-related communication

Mode-related communication shall implement a 1:1 or 1:n scenario but the creation of an n:1 configuration shall be considered invalid. [\]\(RS_SWCT_03200, RS_SWCT_03110\)](#)

As a consequence of [\[TPS_SWCT_01581\]](#), [\[constr_1101\]](#) is formulated.

[constr_1101] Mode-related communication [\[An RPortPrototype typed by ModeSwitchInterface shall not be referenced by more than one SwConnector. \]\(\)](#)

9.1 Declaration of Modes

The SW-Component Template provides some simple means to define collections of modes.

[TPS_SWCT_01071] ModeDeclaration [\[The name of the mode is the most important attribute that has to be provided for each ModeDeclaration. The ModeDeclarations are grouped together within the ModeDeclarationGroup. \]\(RS_SWCT_03200, RS_SWCT_03110\)](#)

[TPS_SWCT_01067] Initial mode [\[The initialMode is active before any mode switches occurred. \]\(RS_SWCT_03200\)](#)

This is shown in Figure [9.1](#)

¹See also AUTOSAR Glossary for "Standardized AUTOSAR Interface".

[TPS_SWCT_01010] defines the meaning of these values.

It is **not allowed** to define any custom or project-specific value of the attribute `ModeDeclarationGroup.category`. $\downarrow()$

As the attributes `ModeDeclaration.value` and `ModeDeclarationGroup.onTransitionValue` are optional the following rule applies:

[constr_1298] Existence of attributes if `category` of a `ModeDeclarationGroup` is set to `EXPLICIT_ORDER` $\downarrow$ The attributes `ModeDeclarationGroup.onTransitionValue` and `ModeDeclaration.value` (for each `ModeDeclaration`) shall be set if the `category` of a `ModeDeclarationGroup` is set to `EXPLICIT_ORDER`. $\downarrow()$

[constr_1299] Existence of attributes if `category` of a `ModeDeclarationGroup` is set to other than `EXPLICIT_ORDER` $\downarrow$ The attributes `ModeDeclarationGroup.onTransitionValue` or `ModeDeclaration.value` (for any `ModeDeclaration`) shall **not** be set if the `category` of a `ModeDeclarationGroup` is set to any value **other than** `EXPLICIT_ORDER`. $\downarrow()$

[constr_1181] Numerical values used in `ModeDeclaration.value` and `ModeDeclarationGroup.onTransitionValue` $\downarrow$ The numerical values used to define the `value` attributes and the `onTransitionValue` attribute of a `ModeDeclarationGroup` shall not overlap. $\downarrow()$

In other words, it is not allowed that the values of two `value` attributes within one `ModeDeclarationGroup` have the same numerical value. Neither is it allowed that the numerical value of the `ModeDeclarationGroup.onTransitionValue` attribute and the numerical value of one of the corresponding `value` attributes are identical.

[TPS_SWCT_01009] The numerical values used to define the values of `ModeDeclaration.value` and `ModeDeclarationGroup.onTransitionValue` can be arbitrarily defined $\downarrow$ As long as the constraints [constr_1181], [constr_1298], and [constr_1299] are fulfilled, the numerical values used to define the values of `ModeDeclaration.value` and `ModeDeclarationGroup.onTransitionValue` can be arbitrarily defined. The numerical values are not required to be consecutive. Gaps are positively allowed. $\downarrow$ (*RS_SWCT_03200*)

Example: the following example of a set of numerical values fulfills all requirements on the definition of `ModeDeclaration.value` and `ModeDeclarationGroup.onTransitionValue`: {1,2, 5, 100}.

Please note that the ability to define `ModeDeclaration.value` and `ModeDeclarationGroup.onTransitionValue` introduces a second heuristics for “ordering” `ModeDeclarations`. If `ModeDeclaration.value` and `ModeDeclarationGroup.onTransitionValue` are not defined the assignment of numerical values to the representations of individual `ModeDeclarations` it is up to the RTE generator to come up with the applicable numerical values.

[TPS_SWCT_01010] `category`s for the definition of a `ModeDeclarationGroup` $\downarrow$ In order to support a clear separation between the two possible ways to influence the

definition of the programmatic representation of `ModeDeclarations` two `categorys` shall be defined for the definition of a `ModeDeclarationGroup`.

- The value of `category` of a `ModeDeclarationGroup` shall be set to `EXPLICIT_ORDER` if it is intended to control the source code generation by means of the values of the attributes `ModeDeclaration.value` and `ModeDeclarationGroup.onTransitionValue`.
- The value of `category` of a `ModeDeclarationGroup` shall be set to `ALPHABETIC_ORDER` if it is intended to let the RTE generator control the source code generation according to the alphabetical sorting.

]([RS_SWCT_03200](#))

More information regarding this aspect can be found in [SWS_Rte_02568].

[TPS_SWCT_01011] Default `category` of a `ModeDeclarationGroup` [For reasons of backwards-compatibility with previous releases of AUTOSAR the default value of the `category` of a `ModeDeclarationGroup` shall be `ALPHABETIC_ORDER`.]([RS_SWCT_03200](#))

Class	ModeDeclaration			
Package	M2::AUTOSARTemplates::CommonStructure::ModeDeclaration			
Note	Declaration of one Mode. The name and semantics of a specific mode is not defined in the meta-model.			
Base	ARObject, AtpClassifier, AtpFeature, AtpStructureElement, Identifiable , Multilanguage Referrable, Referrable			
Attribute	Datatype	Mul.	Kind	Note
value	PositiveInteger	0..1	attr	The RTE shall take the value of this attribute for generating the source code representation of this ModeDeclaration.

Table 9.1: ModeDeclaration

Class	ModeDeclarationGroup			
Package	M2::AUTOSARTemplates::CommonStructure::ModeDeclaration			
Note	A collection of Mode Declarations. Also, the initial mode is explicitly identified. Tags: atp.recommendedPackage=ModeDeclarationGroups			
Base	ARElement , ARObject, AtpBlueprint , AtpBlueprintable, AtpClassifier, Atp Type, CollectableElement, Identifiable , MultilanguageReferrable, Packageable Element, Referrable			
Attribute	Datatype	Mul.	Kind	Note
initialMode	ModeDeclaration	1	ref	The initial mode of the ModeDeclarationGroup. This mode is active before any mode switches occurred.

Attribute	Datatype	Mul.	Kind	Note
modeDeclaration	ModeDeclaration	1..*	aggr	The ModeDeclarations collected in this ModeDeclarationGroup. Stereotypes: atpVariation Tags: vh.latestBindingTime=blueprintDerivationTime
modeManagerErrorBehavior	ModeErrorBehavior	0..1	aggr	This represents the ability to define the error behavior expected by the mode manager in case of errors on the mode user side (e.g. terminated mode user).
modeTransition	ModeTransition	*	aggr	This represents the available ModeTransitions of the ModeDeclarationGroup
modeUserErrorBehavior	ModeErrorBehavior	0..1	aggr	This represents the definition of the error behavior expected by the mode user in case of errors on the mode manager side (e.g. terminated mode manager).
onTransitionValue	PositiveInteger	0..1	attr	The value of this attribute shall be taken into account by the RTE generator for programmatically representing a value used for the transition between two statuses.

Table 9.2: ModeDeclarationGroup

[TPS_SWCT_01450] Semantics of a [ModeTransition](#) [In addition to the ability to specify [ModeDeclarations](#) within a [ModeDeclarationGroup](#) it is also feasible to define possible transitions between [ModeDeclarations](#) within the given [ModeDeclarationGroup](#). This can be done by means of aggregation [ModeTransition](#) at [ModeDeclarationGroup](#) in the role [modeTransition](#). More details are explained in Figure 9.2.]([RS_SWCT_03200](#))

[TPS_SWCT_01451] Relations between [ModeTransition](#) and [ModeDeclaration](#) [[ModeTransition](#) has two associations with the multiplicity 1 to [ModeDeclaration](#):

- The reference [enteredMode](#) denotes a [ModeDeclaration](#) that can be entered as part of the enclosing [ModeTransition](#).
- The reference [exitedMode](#) denotes a [ModeDeclaration](#) that can be exited as part of the enclosing [ModeTransition](#).

] ([RS_SWCT_03200](#))

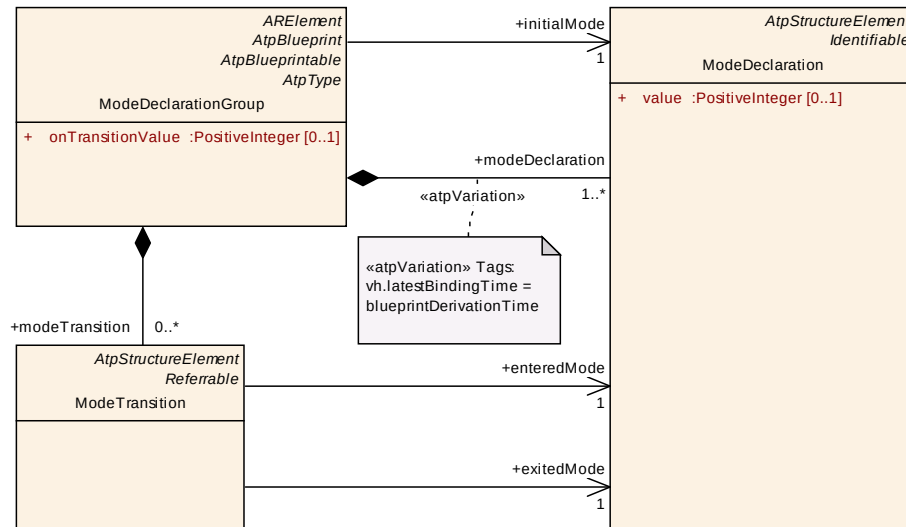


Figure 9.2: ModeTransition

[constr_1193] **ModeDeclaration** shall be referenced by at least one **ModeTransition** in the role **enteredMode** [For each **ModeDeclaration** at least one **ModeTransition** shall reference the **ModeDeclaration** in the role **enteredMode**. This constraint shall apply **only** if there is at least one **ModeTransition** defined in the context of the enclosing **ModeDeclarationGroup** and it shall **not** apply to the **initialMode**.]()

For clarification, the **ModeDeclarationGroup.initialMode** does not need to be referenced by an **enteredMode** because by identifying this **ModeDeclaration** in the role **initialMode** it is clear that the **ModeDeclaration** will be entered at least once.

Class	ModeTransition			
Package	M2::AUTOSARTemplates::CommonStructure::ModeDeclaration			
Note	This meta-class represents the ability to describe possible ModeTransitions in the context of a ModeDeclarationGroup.			
Base	ARObject, AtpClassifier, AtpFeature, AtpStructureElement, Identifiable, Multilanguage Referrable, Referrable			
Attribute	Datatype	Mul.	Kind	Note
enteredMode	ModeDeclaration	1	ref	This represents the entered model of the ModeTransition.
exitedMode	ModeDeclaration	1	ref	This represents the exited mode of the ModeTransition

Table 9.3: ModeTransition

9.2 Modes and Events

[TPS_SWCT_01376] **Software-components need to be capable of reacting to state changes** [Software-components need to be capable of reacting to state

changes issued by some Mode Manager and adopt their behavior to the new situation.](RS_SWCT_03110)

Such a mode dependent software-component is shown in Figure 9.3.

[TPS_SWCT_01077] Configure the response to mode changes [Since the behavior of `AtomicSwComponentTypes` is mainly determined by the `RunnableEntities` contained in the `SwcInternalBehavior` it is necessary to configure the response to mode changes on the level of `RunnableEntities`.](RS_SWCT_03120)

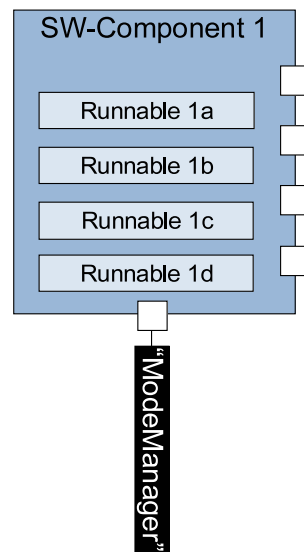


Figure 9.3: State Managers and software-components

Figure 9.4 shows an excerpt of the meta-model illustrating how the relationship between the current mode and the `SwcInternalBehavior` of the `AtomicSwComponentType` can be described.

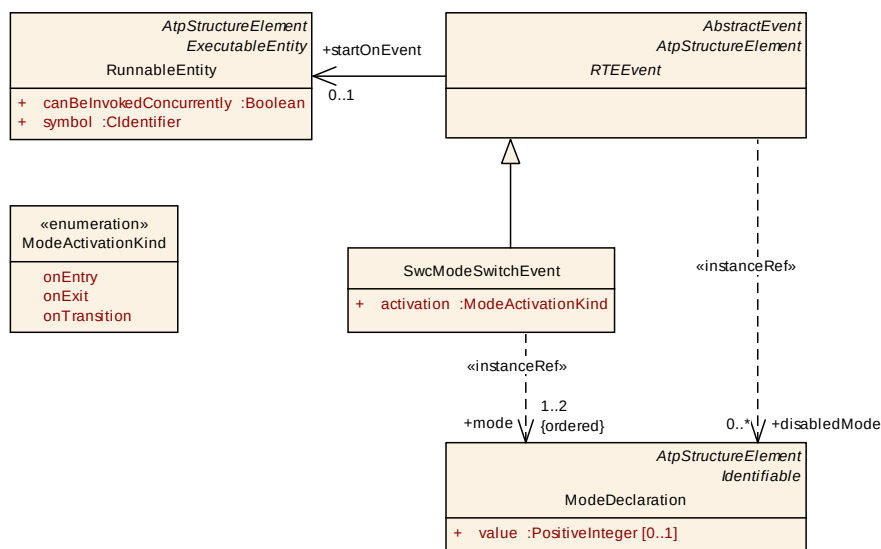


Figure 9.4: Modes and events

[TPS_SWCT_01377] Two mechanisms to define how `SwcInternalBehavior` should interact with the mode management [A `AtomicSwComponentType` can use two mechanisms to define how its `SwcInternalBehavior` should interact with the mode management. Both mechanisms are visible in Figure 9.4.]([RS_SWCT_03110](#))

[TPS_SWCT_01378] `AtomicSwComponentType` can define an `SwcModeSwitchEvent` to execute `RunnableEntity` [Using the first mechanism, an `AtomicSwComponentType` can define an `SwcModeSwitchEvent` to specify that a particular `RunnableEntity` shall be started whenever a mode is entered, exited, or a transition between two specified modes occurs.]([RS_SWCT_03110](#))

[constr_4003] Semantics of `SwcModeSwitchEvent` [If the value of `SwcModeSwitchEvent.activation` is `onTransition` then `SwcModeSwitchEvent` shall refer to two different `ModeDeclarations` belonging to the same instance of `ModeDeclarationGroup`.

Their order defines the direction of the transition from one mode into another. In all other cases `SwcModeSwitchEvent` shall refer to exactly one `ModeDeclaration`.]()

[constr_1195] `SwcModeSwitchEvent` and the definition of `ModeTransition` [For each pair of `ModeDeclarations` referenced by a `SwcModeSwitchEvent` with attribute `activation` set to `onTransition` a `ModeTransition` shall be defined in the corresponding direction (i.e. from `exitedMode` to `enteredMode`). This constraint shall only apply if the respective `ModeDeclarationGroup` defines at least one `modeTransition`.]()

[TPS_SWCT_01379] `AtomicSwComponentType` can indicate whether an `RTEEvent` that starts an associated `RunnableEntity` is disabled in a certain mode [Using the second mechanism, the `AtomicSwComponentType` can indicate whether an `RTEEvent` that starts an associated `RunnableEntity` is disabled in a certain mode.

That is, `RTEEvents` without an association in the role `disabledMode` are processed regularly according to their definition.

`RTEEvents` with the optional association `disabledMode` have the additional limitation that the associated `RunnableEntity` is *not* started when the `ModeDeclaration` referenced as `disabledMode` is active.]([RS_SWCT_03110](#))

The mechanisms discussed so far have to be applied for the `SwcInternalBehavior` on the receiver side of mode switches. Since mode switches are received via `PortPrototypes` the following constraints apply:

[TPS_SWCT_01380] Mode management behavior on the sender side [On the sender side, a `RunnableEntity` shall have `ModeSwitchPoints` that eventually associate a `RunnableEntity` with the specific `ModeDeclarationGroups` which it manages, see Figure 9.5.]([RS_SWCT_03110](#))

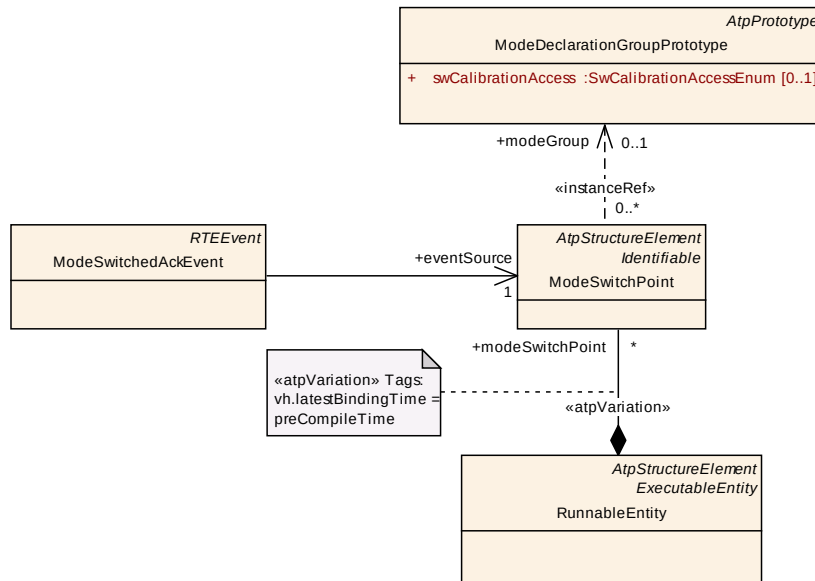


Figure 9.5: **ModeSwitchPoint**

Class	ModeSwitchPoint			
Package	M2::AUTOSAR Templates::SWComponentTemplate::SwcInternalBehavior::Mode DeclarationGroup			
Note	A ModeSwitchPoint is required by a RunnableEntity owned a Mode Manager. Its semantics implies the ability to initiate a mode switch.			
Base	ARObject, AtpClassifier, AtpFeature, AtpStructureElement, Identifiable, Multilanguage Referrable, Referrable			
Attribute	Datatype	Mul.	Kind	Note
modeGroup	ModeDeclarationGroupPrototype	0..1	iref	The mode declaration group that is switched by this runnable.

Table 9.4: **ModeSwitchPoint**

[TPS_SWCT_01383] **ModeSwitchPoint** [The **ModeSwitchPoint** also allows for the definition of a **ModeSwitchedAckEvent** if this is requested by the definition of the **PPortPrototype** (see also 4.5.3). This **RTEEvent** is eventually owned by a mode manager to allow for getting confirmation of a mode change.]([RS_SWCT_03110](#))

[TPS_SWCT_01555] **ModeSwitchedAckEvent** is triggered by the RTE regardless [The **ModeSwitchedAckEvent** is triggered by the RTE (for more details please refer to [2]) regardless which **RunnableEntity** has requested the mode switch notification, even if the Meta Model implies a reference from **ModeSwitchedAckEvent** to a specific **ModeSwitchPoint** in the role **eventSource**.]([RS_SWCT_03110](#))

[constr_4012] **Timeout of ModeSwitchedAckEvent** [The timeout value of a **Wait-Point** associated with a **ModeSwitchedAckEvent** shall be equal to the corresponding **ModeSwitchedAckRequest.timeout**.]()

Class	ModeSwitchedAckRequest			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
Note	Requests acknowledgements that a mode switch has been proceeded successfully			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
timeout	TimeValue	1	attr	Number of seconds before an error is reported or in case of allowed redundancy, the value is sent again.

Table 9.5: ModeSwitchedAckRequest

Class	ModeSwitchedAckEvent			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::RTE Events			
Note	The event is raised when the referenced modes have been received or an error occurs.			
Base	ARObject, AbstractEvent, AtpClassifier, AtpFeature, AtpStructure Element, Identifiable, MultilanguageReferrable, RTEEvent, Referrable			
Attribute	Datatype	Mul.	Kind	Note
eventSource	ModeSwitchPoint	1	ref	Mode switch point that triggers the event.

Table 9.6: ModeSwitchedAckEvent

[TPS_SWCT_01381] Read the currently active mode [For *Mode Manager* and *Mode User* it might additionally be required to read the currently active mode. For that purpose the a [RunnableEntity](#) that requires read access to the [ModeDeclarationGroupPrototype](#)'s current mode has to define a [ModeAccessPoint](#).]([RS_SWCT_03110](#))

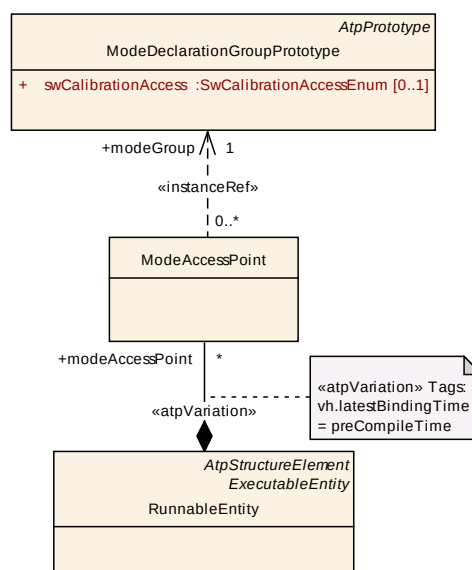


Figure 9.6: ModeAccessPoint

Class	ModeAccessPoint			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::ModeDeclarationGroup			
Note	A ModeAccessPoint is required by a RunnableEntity owned by a Mode Manager or Mode User. Its semantics implies the ability to access the current mode (provided by the RTE) of a ModeDeclarationGroupPrototype's ModeDeclarationGroup.			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
ident	ModeAccessPointIdent	0..1	aggr	The aggregation in the role ident provides the ability to make the ModeAccessPoint identifiable. From the semantical point of view, the ModeAccessPoint is considered a first-class Identifiable and therefore the aggregation in the role ident shall always exist (until it may be possible to let ModeAccessPoint directly inherit from Identifiable). Tags: atp.Status=shallBecomeMandatory xml.sequenceOffset=-100
modeGroup	ModeDeclarationGroupPrototype	0..1	iref	The mode declaration group that is accessed by this runnable. Tags: xml.typeElement=true

Table 9.7: ModeAccessPoint

[TPS_SWCT_01382] Mode switch requests are handled asynchronously by the RTE [Mode switch requests are handled asynchronously by the RTE. Therefore, *Mode Managers* implementation might require to read back the current active mode to synchronize internally to the RTE. A [ModeSwitchPoint](#) does **not** automatically provide read access to the [ModeDeclarationGroupPrototype](#)'s current mode.]([RS_SWCT_03110](#))

[constr_1098] Mode switch and mode disabling [A [SwcModeSwitchEvent](#) shall not simultaneously reference to the same [ModeDeclaration](#) in both the roles [mode](#) and [disabledMode](#).]()

If [\[constr_1098\]](#) would not apply it might happen that a [RunnableEntity](#) would be triggered by a [SwcModeSwitchEvent](#) and on the same time it would be suppressed by the mode disabling.

9.3 Initialization / Finalization

The AUTOSAR standard shall support the execution of initialization code for every [AtomicSwComponentType](#).

[TPS_SWCT_01384] Execution of initialization code for software-components [Most [AtomicSwComponentTypes](#) will need to initialize by executing specific code;

this code shall complete before any other code in the component is executed. Data will be initializing to specific values before the "normal" application software is running. [\]\(RS_SWCT_03110\)](#)

[TPS_SWCT_01385] Execution of finalization code for software-components [Most [AtomicSwComponentTypes](#) will need to finalize by calling specific code; this code shall complete before the functionality of the application software shut down (e.g. a motor drive in a start or end position). [\]\(RS_SWCT_03110\)](#)

[TPS_SWCT_01388] Initial modes of [AtomicSwComponentTypes](#) are defined by the [initialMode](#) [The initial modes of [AtomicSwComponentTypes](#) are defined by the [initialMode](#) references of the required [ModeDeclarationGroups](#). These modes are activated before any other mode activation has occurred. It is the responsibility of the RTE to activate all initial modes on a certain ECU. [\]\(RS_SWCT_03110\)](#)

For more details please refer to the specification of the SWS RTE [2].

9.4 Mode Error Behavior

With the advent of partitions in the AUTOSAR standard, it is important to consider the behavior of mode management with respect to the following scenarios:

- The partition of the mode manager is terminated.
- The partition of the mode user is terminated.

Whenever one of the two scenarios becomes reality, it is important to implement a stable reaction of both mode manager and mode user to the event. In addition, mode manager and mode user should be able to synchronize in terms of which mode shall apply as fast and seamless as possible.

For this purpose, additional modeling support has been defined such that the applicable [ModeDeclarationGroup](#) (which is part of the contract between mode manager and mode user) becomes the place where the policy towards a reaction to e.g. a partition restart is defined.

[TPS_SWCT_01530] Error behavior of mode manager and mode user [The behavior in response to a mode manager getting out of sync with a mode user (because the partition of the mode user has been terminated) or vice versa (because the partition of the mode manager has been terminated) can be defined for the mode manager by means of the attribute [ModeDeclarationGroup.modeManagerErrorBehavior](#) and for the mode user by means of the attribute [ModeDeclarationGroup.modeUserErrorBehavior](#). [\]\(RS_SWCT_03110\)](#)

[TPS_SWCT_01531] The semantics of [ModeErrorReactionPolicyEnum](#) [The attribute [ModeErrorBehavior.errorReactionPolicy](#) shall be used to specify the behavior in the event of a mode error:

[lastMode](#) The last mode applicable before the event shall be assumed.

defaultMode This represents the ability to specify a dedicated mode that shall be made applicable. The identified `ModeDeclaration` could be identical to the `ModeDeclarationGroup.initialMode` but it can just as well be any other `ModeDeclaration` defined in the context of the enclosing `ModeDeclarationGroup`.

⌋(*RS_SWCT_03110*)

[TPS_SWCT_01532] The role of `ModeErrorBehavior.defaultMode` ⌈ The attribute `ModeErrorBehavior.defaultMode` shall be used to identify the particular `ModeDeclaration` if `ModeErrorBehavior.errorReactionPolicy` is set to `defaultMode`. ⌋(*RS_SWCT_03110*)

[constr_1263] Existence of `ModeErrorBehavior.defaultMode` ⌈ The optional attribute `ModeErrorBehavior.defaultMode` **shall exist** if the value of the attribute `ModeErrorBehavior.errorReactionPolicy` is set to `defaultMode`. ⌋()

[TPS_SWCT_01533] `ModeDeclarationGroup.initialMode` shall be assumed in the absence of `ModeDeclarationGroup.modeManagerErrorBehavior` ⌈ If the attribute `ModeDeclarationGroup.modeManagerErrorBehavior` is not defined it shall be assumed that the `ModeDeclarationGroup.initialMode` becomes applicable in case of the mode manager getting out of sync with a mode user (because the partition of the mode user has been terminated). ⌋(*RS_SWCT_03110*)

[TPS_SWCT_01534] `ModeDeclarationGroup.initialMode` shall be assumed in the absence of `ModeDeclarationGroup.modeUserErrorBehavior` ⌈ If the attribute `ModeDeclarationGroup.modeUserErrorBehavior` is not defined it shall be assumed that the `ModeDeclarationGroup.initialMode` becomes applicable in case of the mode user getting out of sync with a mode manager (because the partition of the mode manager has been terminated). ⌋(*RS_SWCT_03110*)

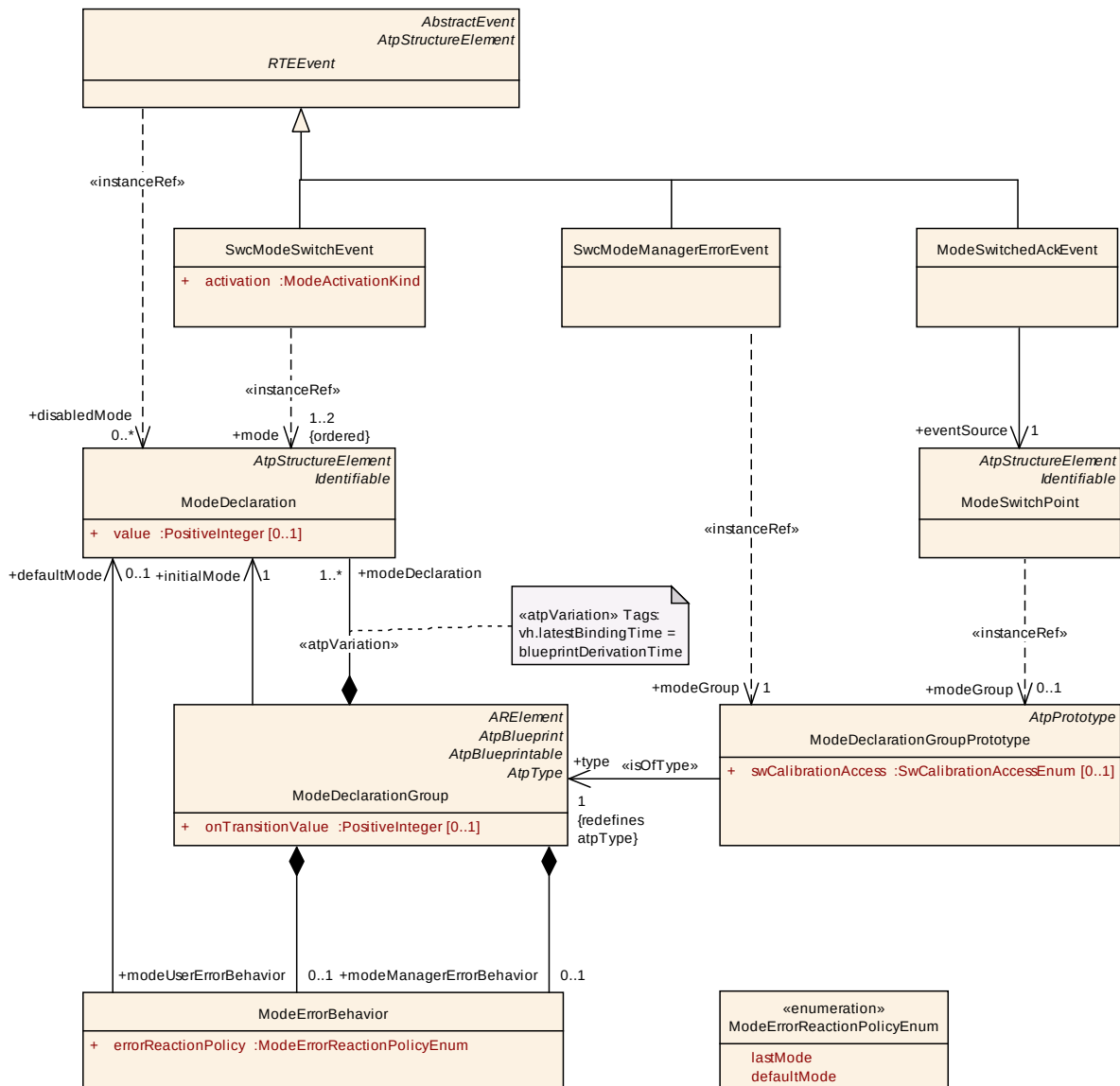


Figure 9.7: Mode Error Behavior

Class	ModeErrorBehavior			
Package	M2::AUTOSARTemplates::CommonStructure::ModeDeclaration			
Note	This represents the ability to define the error behavior in the context of mode handling.			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
defaultMode	ModeDeclaration	0..1	ref	This represents the ModeDeclaration that is considered the error mode in the context of the enclosing ModeDeclarationGroup.
errorReactionPolicy	ModeErrorReactionPolicyEnum	1	attr	This represents the ability to define the policy in terms of which default model shall apply in case an error occurs.

Table 9.8: ModeErrorBehavior

Enumeration	ModeErrorReactionPolicyEnum
Package	M2::AUTOSARTemplates::CommonStructure::ModeDeclaration
Note	This represents the ability to specify the reaction on a mode error.
Literal	Description
defaultMode	This represents the ability to switch to the defaultMode in case of a mode error.
lastMode	This represents the ability to keep the last mode in case of a mode error.

Table 9.9: ModeErrorReactionPolicyEnum

[TPS_SWCT_01535] Mode manager reacts on mode error [If the mode manager is getting out of sync with a mode user (because the partition of the mode user has been terminated) or vice versa (because the partition of the mode manager has been terminated) it shall be possible for the mode manager to react on such an event.

For this purpose the formal [SwcModeManagerErrorEvent](#) is defined that can be taken to e.g. trigger the execution of a [RunnableEntity](#) in response to an error with respect to mode switch communication.]([RS_SWCT_03110](#))

Class	SwcModeManagerErrorEvent			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::RTE Events			
Note	This represents the ability to react on errors occurring during mode handling.			
Base	ARObject, AbstractEvent ,AtpClassifier,AtpFeature,AtpStructure Element, Identifiable ,MultilanguageReferrable, RTEEvent , Referrable			
Attribute	Datatype	Mul.	Kind	Note
modeGroup	ModeDeclarationGroupPrototype	1	iref	This represents the ModeDeclarationGroupPrototype for which the error behavior of the mode manager applies.

Table 9.10: SwcModeManagerErrorEvent

As mentioned in [[constr_1075](#)], it is possible to overrule the default compatibility rules by the definition of a [PortInterfaceMapping](#).

In this case the demand for having identical definitions of [ModeDeclarationGroup.modeUserErrorBehavior](#) and [ModeDeclarationGroup.modeManagerErrorBehavior](#) is no longer valid.

However, there is one additional caveat to observe in this case. This affects the implementation of error behavior in case that several mode users are connected to a mode manager.

[TPS_SWCT_01536] Coherent behavior of all mode users in case of errors in the mode switch communication [The behavior in case of errors with the communication of mode switches needs to be **coherent for all** connected mode users **especially** if the individual [SwConnectors](#) are legitimized by the existence of a [PortInterfaceMapping](#).]([RS_SWCT_03110](#))

[TPS_SWCT_01541] Preferential selection of `modeUserErrorBehavior` [The definition of mode error behavior on the provided side of shall be considered **dominant** over the definition of mode error behavior on the required side.

This means that a `ModeSwitchInterface.modeGroup.type.modeUserErrorBehavior` used to type an `AbstractProvidedPortPrototype` shall be considered **dominant** over the definition of a corresponding `modeUserErrorBehavior` and defined in the context of an `AbstractRequiredPortPrototype`.
]([RS_SWCT_03110](#))

[TPS_SWCT_01542] Preferential selection of `modeManagerErrorBehavior` [The definition of mode error behavior on the provided side of shall be considered **dominant** over the definition of mode error behavior on the required side.

This means that a `ModeSwitchInterface.modeGroup.type.modeManagerErrorBehavior` used to type an `AbstractProvidedPortPrototype` shall be considered **dominant** over the definition of a corresponding `modeManagerErrorBehavior` defined in the context of an `AbstractRequiredPortPrototype`.
]([RS_SWCT_03110](#))

The consequence of [\[TPS_SWCT_01541\]](#) and [\[TPS_SWCT_01542\]](#) is that the **mode manager shall be considered the master of the definition of mode error behavior**.

Please note that the statements made in [\[TPS_SWCT_01541\]](#) is further underlined by [\[SWS_Rte_06795\]](#) and the statement made by [\[TPS_SWCT_01542\]](#) is further underlined by [\[SWS_Rte_06795\]](#).

The details of how the run-time behavior of mode manager and mode user shall look like in the event of the mode manager getting out of sync with a mode user (because the partition of the mode user has been terminated) or vice versa (because the partition of the mode manager has been terminated) as well as the applicable RTE APIs are explained in [\[2\]](#).

9.5 Summary Meta-Model Excerpt Related to Modes

Figure [9.8](#) provides an overview of all meta-model elements that have a direct relationship to the meta-classes involved in the modelling of mode switches.

To get the complete picture, it should be noted that also the concepts of `PortGroups` (see [4.6](#)) and `ServiceProxySwComponentType` (see [11.4](#)) have a semantical relationship to mode management, though this is not expressed via relations in the meta-model.

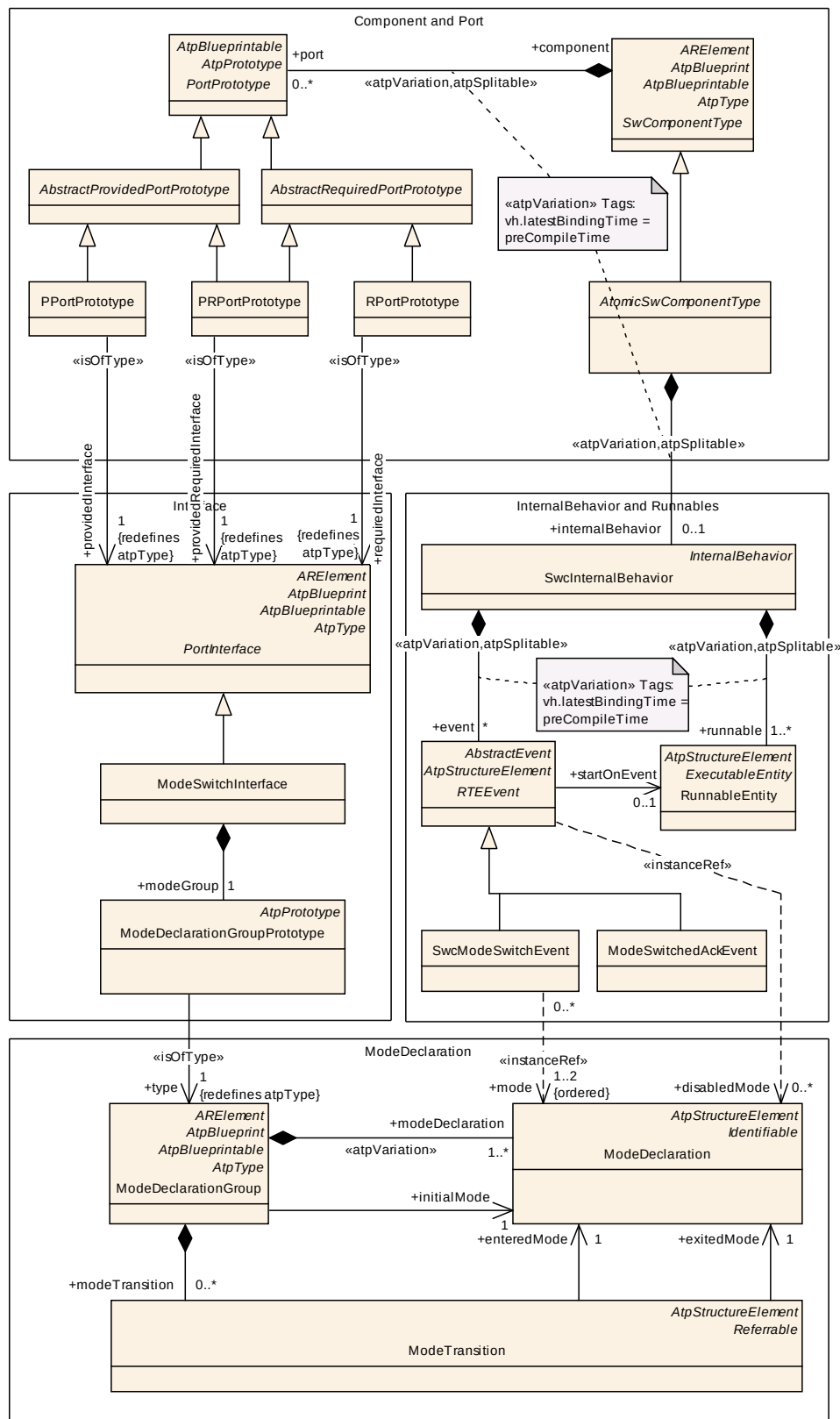


Figure 9.8: Summary meta-model excerpt related to modes

10 ECU Abstraction and Complex Drivers

10.1 Introduction

During the design of embedded systems there is one crucial point where the hardware and software have to be related. In AUTOSAR the `ECU Resource Template` describes the provided hardware resources.

On the other hand, the `Software Component Template` describes software generally without specific hardware in mind. But there are some places where both have to meet and fit.

One interface between hardware and software is discussed in the memory and execution time section of [7]. In this chapter the overall system view of the interface between sensors/actuators and software is described and the consequences for the `Software Component Template` are derived.

10.2 High Level Hardware and Software Architecture

The AUTOSAR concept defines a software architecture (see Figure 10.1) and within this layered architecture the interfaces between the hardware and the software are explicitly modeled.

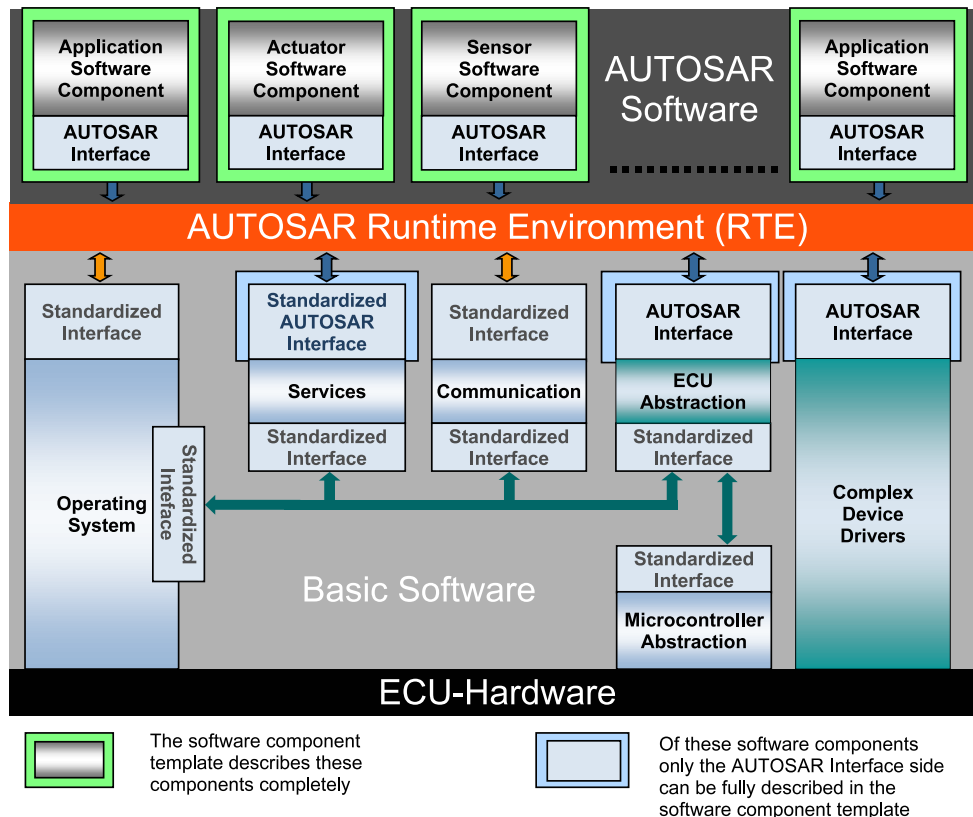


Figure 10.1: AUTOSAR ECU Software Architecture

The signal¹ flow from a hardware to software and vice versa will be described in the following sections.

A sensor² is converting a physical value (1) in Figure 10.2 (e.g. temperature, force, light intensity) into an electrical signal (2) which can be either a current or a voltage.

Inside the ECU generally there will be some electronics to enhance the electrical signal provided by the sensor. In AUTOSAR this is called ECU Electronics. This electronics is also responsible for the conversion of the electrical signal into a microcontroller compatible form (3), usually a voltage.

After the electrical signal has been enhanced and converted it will be captured by the microcontroller. This can either be done by a simple digital input, an analogue to digital converter or maybe a pulse-width demodulation module. Now the electrical signal is available as a software data value (4).

This signal flow is sketched in the top part of Figure 10.2.

¹The term “signal” is not going to be used here at its own but more specific terms will be used for the different abstractions of signals at the different stages of the signal flow.

²For the sake of simplicity this discussion is limited to the sensor aspects. Nevertheless, the same applies also for actuators.

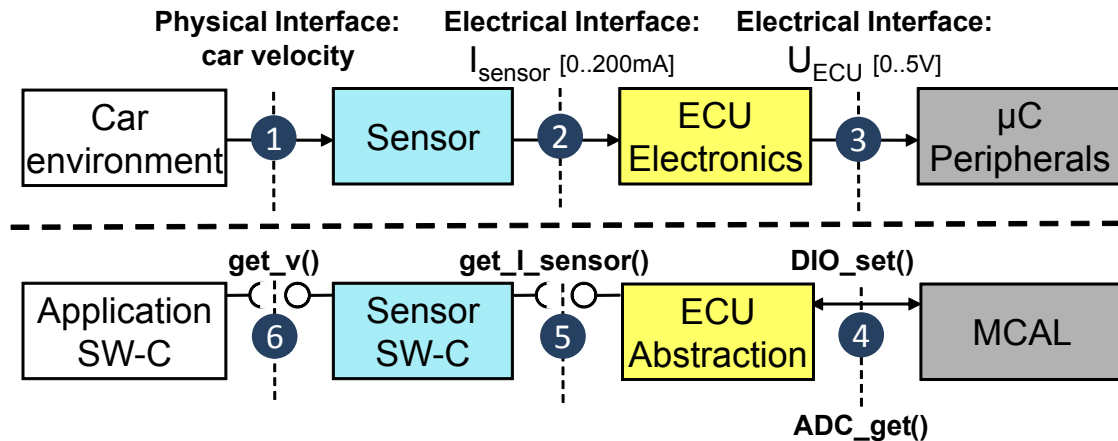


Figure 10.2: Interfaces between hardware and software

This signal chain is represented one to one in the AUTOSAR software architecture and depicted in the lower part of Figure 10.2.

In an implementation of AUTOSAR only the Microcontroller Abstraction (MCAL) has direct access to the peripheral hardware. This layer is going to be standardized and all hardware access should go through this layer. The idea of the AUTOSAR signal flow is to map the hardware to the corresponding software modules.

So if an electrical current is the input to the microcontroller peripheral, the MCAL will deliver a data value that represents this current. As the ECU Electronics has enhanced and converted the electrical signal prior to the microcontroller, the corresponding software entity is reversing this conversion. This is performed in the ECU Abstraction layer.

So if the input to the ECU is an electrical current and the ECU Electronics has converted this current into a voltage (from 2 to 3), the ECU Abstraction will convert the data value voltage into an AUTOSAR signal representing a current (from 4 to 5). This AUTOSAR signal represents the actual current that was provided by the sensor (2).

Now the first step in the conversion has to be reversed: the sensor has converted a physical value into an electrical signal. And so the Sensor Software Component has to reverse this again. The Sensor Software Component will read the AUTOSAR signal representing the electrical value and transform it into an AUTOSAR signal representation of the physical value (from 5 to 6).

Now this physical value is available on the RTE and can be consumed or read by other SW-Components. Although the interface between the ECU Abstraction and the Sensor Software Component is also an AUTOSAR interface and could be routed through some communication bus, it will not be practical to separate the ECU Abstraction and the corresponding [SensorActuatorSwComponentType](#) due to potentially high communication effort.

In Figure 10.3 a complete signal flow from a sensor input to an actuator output is shown.

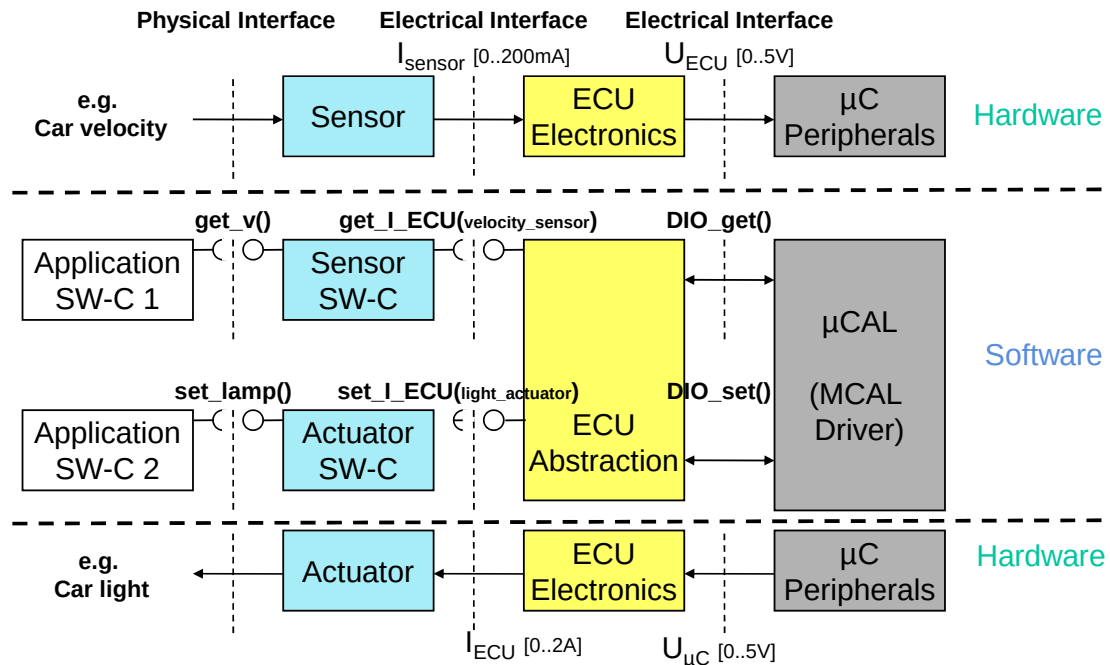


Figure 10.3: Sensor and Actuator Signal Flow

In the next section the interfaces between the involved software modules are discussed.

10.3 Interfaces and APIs

Two fundamentally different interfaces are involved when converting from sensors/actuators to software components, see markers “4” and “5” in Figure 10.2.

The interface between the Microcontroller Abstraction and the ECU Abstraction is a Standardized Interface (see AUTOSAR Glossary [42]). This interface is not visible on the Virtual Function Bus and therefore the MCAL and ECU Abstraction have to be present on the same ECU.

For further description of this interface please refer to the ECU Resource Template documentation.

The interface to the [SensorActuatorSwComponentTypes](#) is visible on the Virtual Function Bus. In general the [SensorActuatorSwComponentType](#) should be on the same ECU as the ECU hardware abstraction.

Also the interface between the [SensorActuatorSwComponentTypes](#) and the actual [AtomicSwComponentTypes](#) representing the application is visible on the VFB. To describe the data that is going to be exchanged via this interface the standard AUTOSAR Interface description mechanisms are used (see chapter 3.4).

10.3.1 ECU Abstraction and its AUTOSAR Interfaces

Since the AUTOSAR standard is designed with the focus on the integration of software-components coming from different contractors, the interfaces between the different software-components obviously have to be compatible.

In the case of the sensors and actuators the interface is gathered in the ECU Abstraction. For each sensor and actuator there is one AUTOSAR [PortPrototype](#) that represents the AUTOSAR Signal that is delivered by the sensor or the AUTOSAR Signal that is consumed by the actuator. This relationship is depicted in Figure 10.4

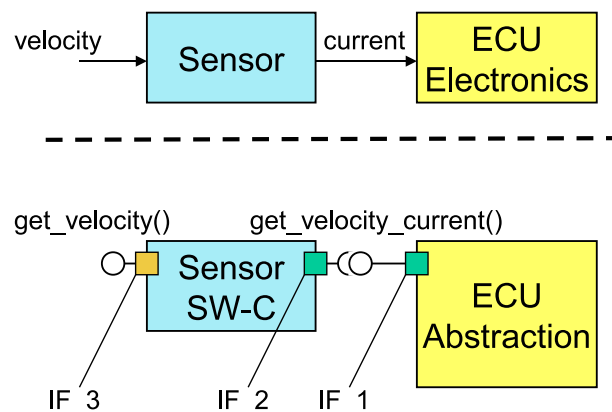


Figure 10.4: Interfaces of signals in software

Each sensor and actuator has an AUTOSAR [PortPrototype](#) at the ECU Abstraction. Connected to this port is the [SensorActuatorSwComponentType](#). The [SensorActuatorSwComponentType](#) has one [PortPrototype](#) (i.e. IF_2) to the ECU Abstraction (which provides the values via IF_1) where it gets the AUTOSAR signals from the hardware, and one [PortPrototype](#) (i.e. IF_3) to [AtomicSwComponentTypes](#) where it provides the actual physical value to the rest of AUTOSAR on the RTE.

In addition, the Interfaces between the ECU Abstraction and the [SensorActuatorSwComponentType](#) have to be compatible like defined in chapter 6.

10.4 Sensors/Actuators

In the layered software architecture described in [6] each hardware sensor/actuator is coupled to a [SensorActuatorSwComponentType](#) (see Figure 10.5).

[TPS_SWCT_01047] Reference from the software representation of a sensor/actuator to the actual hardware element | Since the Software Component Template is going to be used to describe the [SensorActuatorSwComponentType](#) as well, there is also a reference needed from the software representation of a sensor/actuator to the actual hardware element described in the ECU Resource description. |(RS_SWCT_02080, RS_SWCT_03090)

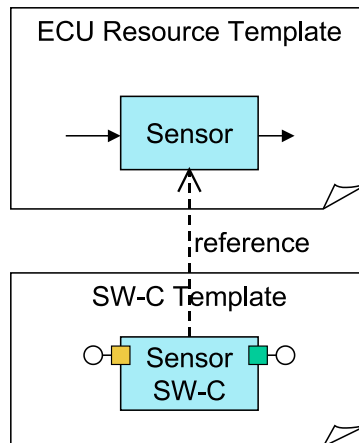


Figure 10.5: Shipment of a sensor

So each time a sensor/actuator is selected to be connected to an ECU also the corresponding `SensorActuatorSwComponentType` is available.

[constr_1144] `SensorActuatorSwComponentType`, `EcuAbstractionSwComponentType`, and `ComplexDeviceDriverSwComponentType` may only reference a `HwType` [The attribute `sensorActuator` of `SensorActuatorSwComponentType`, the attribute `hardwareElement` of `EcuAbstractionSwComponentType`, and the attribute `hardwareElement` of `ComplexDeviceDriverSwComponentType` may only reference a `HwType`. References to other subclasses of `HwDescriptionEntity` are not allowed.]()

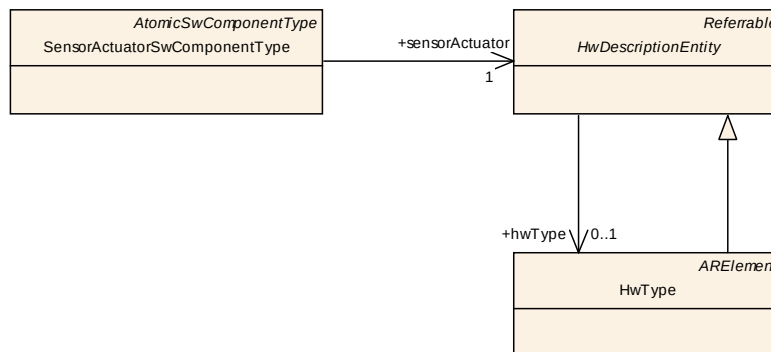


Figure 10.6: Sensor/actuator to Hardware Relationship

Figure 10.6 depicts the reference of `SensorActuatorSwComponentType` designed as a specialization of an `AtomicSwComponentType` with an additional reference to a `HwType`.

[constr_1109] Mapping of `SwComponentPrototypes` typed by a `SensorActuatorSwComponentType` [A `SwComponentPrototype` typed by a `SensorActuatorSwComponentType` needs to be mapped and run on exactly that ECU that contains the `HwElement` corresponding to the `HwType` that its `SensorActuatorSwComponentType` refers to in case it accesses the hardware via the I/O hardware abstraction layer.]()

[TPS_SWCT_01048] [SensorActuatorSwComponentType](#) may use the I/O hardware abstraction directly [In contrast to an [ApplicationSwComponentType](#), a [SensorActuatorSwComponentType](#) may use the I/O hardware abstraction directly (via ports/connectors).] ([RS_SWCT_02080](#), [RS_SWCT_03090](#))

In case the sensor/actuator hardware is accessed via bus communication, e.g. is located on a LIN slave, no such mapping constraints apply (note that this is not handled via the IO hardware abstraction layer).

Class	SensorActuatorSwComponentType			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	The SensorActuatorSwComponentType introduces the possibility to link from the software representation of a sensor/actuator to its hardware description provided by the ECU Resource Template. Tags: atp.recommendedPackage=SwComponentTypes			
Base	ARElement , ARObject , AtomicSwComponentType , AtpBlueprint , AtpBlueprintable , AtpClassifier , AtpType , CollectableElement , Identifiable , MultilanguageReferrable , PackageableElement , Referrable , SwComponentType			
Attribute	Datatype	Mul.	Kind	Note
sensorActuator	HwDescriptionEntity	1	ref	Reference from the Sensor Actuator Software Component Type to the description of the actual hardware.

Table 10.1: SensorActuatorSwComponentType

10.5 I/O Hardware Abstraction

[TPS_SWCT_01389] I/O Hardware Abstraction interfaces MCAL drivers [The I/O Hardware Abstraction interfaces on one side the MCAL drivers via Standardized Interfaces and on the other side the Sensor Actuator Software Component via AUTOSAR Interfaces. On the VFB[3] the I/O Hardware Abstraction is represented by the [EcuAbstractionSwComponentType](#).]()

[TPS_SWCT_01390] I/O Hardware Abstraction might have sub-structures [Depending on the complexity of an ECU, the I/O Hardware Abstraction might have sub-structures. In this case the I/O Hardware Abstraction Layer is described by several different [EcuAbstractionSwComponentTypes](#) on M1.]()

Class	EcuAbstractionSwComponentType			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	The ECUAbstraction is a special AtomicSwComponentType that resides between a software-component that wants to access ECU periphery and the Microcontroller Abstraction. The EcuAbstractionSwComponentType introduces the possibility to link from the software representation to its hardware description provided by the ECU Resource Template. Tags: atp.recommendedPackage=SwComponentTypes			
Base	ARElement , ARObject , AtomicSwComponentType , AtpBlueprint , AtpBlueprintable , AtpClassifier , AtpType , CollectableElement , Identifiable , MultilanguageReferrable , PackageableElement , Referrable , SwComponentType			
Attribute	Datatype	Mul.	Kind	Note
hardwareElement	HwDescriptionEntity	*	ref	Reference from the EcuAbstractionComponentType to the description of the used HwElements.

Table 10.2: EcuAbstractionSwComponentType

[TPS_SWCT_01391] I/O Hardware Abstraction abstracts from the location of peripheral I/O devices [The I/O Hardware Abstraction abstracts from the location of peripheral I/O devices (on-chip or on-board) and the ECU hardware layout and has therefore dependencies to ECU Hardware described by [HwElements](#). In addition, the [EcuAbstractionSwComponentType](#) is a hybrid concept sharing features of both software-components and basic software modules.]()

[TPS_SWCT_01392] Mapping between the [EcuAbstractionSwComponentType](#) and the corresponding [BswModuleDescription](#) [The BSW part is described by the means of the Basic Software Module Template. The mapping between the [EcuAbstractionSwComponentType](#) and the corresponding [BswModuleDescription](#) is provided by the class [SwcBswMapping](#) which in addition also maps the two corresponding [InternalBehaviors](#). This mechanism is further explained in [7].]()

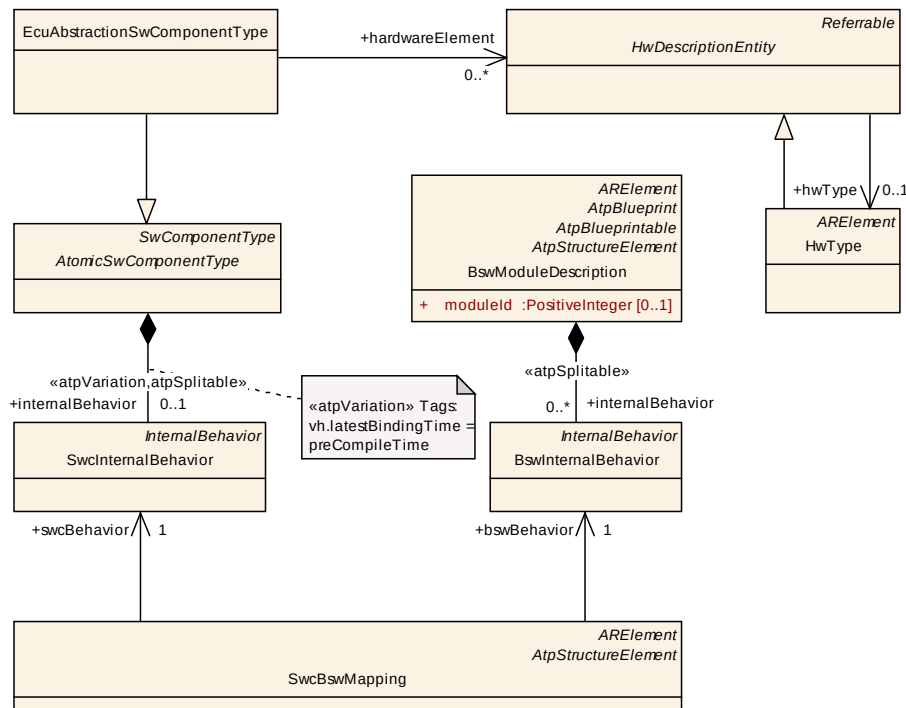


Figure 10.7: **EcuAbstractionSwComponentType**

10.6 Complex Driver

[TPS_SWCT_01393] Complex Driver [A **Complex Driver** implements complex sensor evaluation and actuator control with direct access to the Microcontroller using specific interrupts and/or complex Microcontroller peripherals to fulfill the special functional and timing requirements.

In addition it might be used to implement enhanced services / protocols or encapsulates legacy functionality of a **non-AUTOSAR system**.]()

See also document [3].

[TPS_SWCT_01394] Complex Driver is represented by the ComplexDeviceDriverSwComponentType [On the VFB the **Complex Driver** is represented by the **ComplexDeviceDriverSwComponentType**. An ECU might have zero to many different **ComplexDeviceDriverSwComponentTypes**.]()

Class	ComplexDeviceDriverSwComponentType			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	The ComplexDeviceDriverSwComponentType is a special AtomicSwComponentType that has direct access to hardware on an ECU and which is therefore linked to a specific ECU or specific hardware. The ComplexDeviceDriverSwComponentType introduces the possibility to link from the software representation to its hardware description provided by the ECU Resource Template. Tags: atp.recommendedPackage=SwComponentTypes			
Base	ARElement , ARObject , AtomicSwComponentType , AtpBlueprint , AtpBlueprintable , AtpClassifier , AtpType , CollectableElement , Identifiable , MultilanguageReferrable , PackageableElement , Referrable , SwComponentType			
Attribute	Datatype	Mul.	Kind	Note
hardwareElement	HwDescriptionEntity	*	ref	Reference from the ComplexDeviceDriverSwComponentType to the description of the used HwElements.

Table 10.3: ComplexDeviceDriverSwComponentType

[TPS_SWCT_01395] **ComplexDeviceDriverSwComponentType** has dependencies to ECU Hardware [Similar to [EcuAbstractionSwComponentType](#) the [ComplexDeviceDriverSwComponentType](#) has dependencies to ECU Hardware described by [HwElements](#) and is a hybrid between Software Component and Basic Software Module.]()

[TPS_SWCT_01396] Mapping between the **ComplexDeviceDriverSwComponentType** and the corresponding **BswModuleDescription** [The BSW part is described by the means of the Basic Software Module Template. The mapping between the [ComplexDeviceDriverSwComponentType](#) and the corresponding [BswModuleDescription](#) is provided by the class [SwcBswMapping](#) which in addition also maps the two corresponding [InternalBehaviors](#). This mechanism is further explained in [7].]()

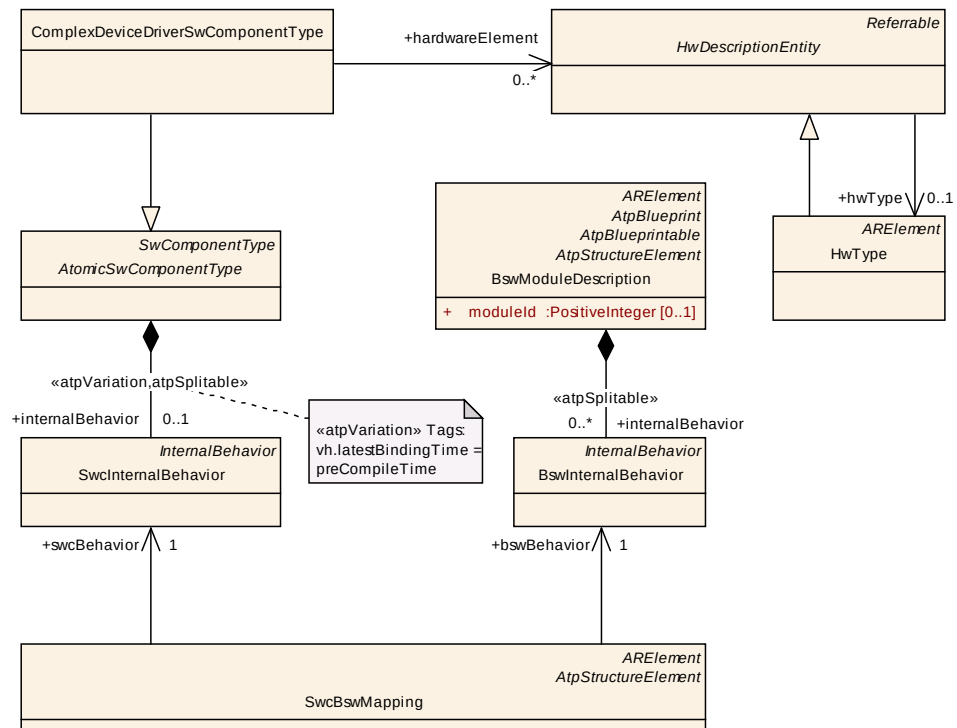


Figure 10.8: **ComplexDeviceDriverSwComponentType**

11 Services

11.1 Overview: Generation of Service-related Model Elements

This chapter covers the description and handling of AUTOSAR `Service` configuration.

[TPS_SWCT_01397] Hybrid concept between Basic Software Modules and a `SwComponentType` `AUTOSAR Services` can be seen as a hybrid concept between Basic Software Modules and a `SwComponentType`. AUTOSAR Services actually provide access to low-level and ECU-wide “standard functionalities” commonly referred to as “service”.

`AtomicSwComponentTypes` that require AUTOSAR `Services` use Standardized AUTOSAR `Interfaces` to communicate with these. The connection of `PortPrototypes` of the `ServiceSwComponentTypes` and `PortPrototypes` of the `AtomicSwComponentTypes` implement several communication patterns. `]()`

I	II	III	IV
A	1:n	PPort : RPort	distribution of data or modes to n SW-Cs, e.g. used for ECU mode
A*	1:n	RPort : PPort	currently not used, not supported for client-server communication
B	1:1	PPort : RPort	SW-C acts as Server, used for so called “call-backs”
B	1:1	RPort : PPort	Service acts as Server, typical Service usage
C*	n:1	PPort : RPort	conceptually not used to support index abstraction via <code>PortDefinedArgumentValues</code>
C	n:1	RPort:PPort	SW-C acts as Server, used for so called “call-backs” invoked by more than one Service

Table 11.1: ServiceConnectorPattern

Legend for Table 11.1:

I Pattern name

II Communication pattern (client/server, sender/receiver)

III Kind of PortPrototype at service : software-component

IV Description, use case

[TPS_SWCT_01398] Communication patterns for AUTOSAR services `]()` The communication patterns for AUTOSAR services are summarized in Table 11.1. `]()`

[TPS_SWCT_01403] Impact of AUTOSAR services on the methodology `]()` Due to this special nature, such `AUTOSAR Services` need to be handled with particular attention in the methodology [4]. That is, a number of elements need to be generated during ECU integration. `]()`

The following list of paragraphs presents a short overview over the steps required for the configuration of `AUTOSAR Services`.

Note that most of these steps are performed by tools and the model elements being created in these steps are rather specific to `Service` configuration and are not to be modeled manually within AUTOSAR authoring tools.

In particular, the following requirements apply:

- **[TPS_SWCT_01399] Dependency is modeled by aggregating required and provided `PortPrototypes`** [The dependency of an `AtomicSwComponentType` (or more precisely, one of its non-abstract derived meta-classes) from an AUTOSAR `Service` is modeled by aggregating required and provided `PortPrototypes`.]()

[TPS_SWCT_01400] `PortInterface` selected from the set of standardized `Service Interfaces` [The `PortInterface` being implemented by the `PortPrototypes` needs to be one of a number of standardized `Service Interfaces` which is indicated by having its `isService` attribute set to `true` and is (via several levels of indirection) finally referenced by `ServiceNeeds`.]()

Additionally, the software components and Basic Software Modules shall specify `ServiceNeeds` containing further input information for the later `Service` configuration step.

- **[TPS_SWCT_01401] Form a top-level `RootSwCompositionPrototype`** [When defining a software system, the `AtomicSwComponentType` is used in the form of `SwComponentPrototypes` within a `CompositionSwComponentType`. In this step, the non-service ports of all required interfaces are being connected using `AssemblySwConnectors` and `DelegationSwConnectors` in order to eventually form a top-level `RootSwCompositionPrototype` which can be referenced in an AUTOSAR `System`.]()
- **[TPS_SWCT_01402] Mapping of all `AtomicSwComponentType` instances to `EcuInstances`** [In System Configuration Phase, the mapping of all `AtomicSwComponentType` instances to `EcuInstances` is done (for the specification of `EcuInstance` see [11]). The `ServiceNeeds` may be used by tools to check for available resources on the targeted ECUs.]()
- **[TPS_SWCT_01404] Creation of the `Ecu Extract`** [The `ECU Extract` is extracted from the System Configuration for each ECU. As explained in the AUTOSAR System Template [11], this contains an ECU-centric view onto the system description.

This includes a reduced version of the system's `RootSwCompositionPrototype` where `SwComponentPrototypes` not being mapped to the ECU are being left out and all `Compositions` are stripped off, so that in the `ECU Extract` only one instance of `CompositionSwComponentType` remains which aggregates all `SwComponentPrototypes` on the ECU in a flat manner.]()

- **[TPS_SWCT_01405] Creation of the `ServiceSwComponentTypes`** [In ECU Configuration, for each `Service` required on the ECU exactly one `ServiceSwComponentType` is created based on the needs from the `AtomicSwComponentTypes`: An adequate number of `PortPrototypes` are created on this `ServiceSwComponentType` for each needed port at the `AtomicSwComponentType`.

Thereby the specified communication pattern A, B or C for a specific kind of `ServicePort` has to be considered. See also chapter 11.3 and table 11.1. `]()`

- **[TPS_SWCT_01406] Creation of `SwComponentPrototype` typed by a `ServiceSwComponentType`** `[` Per `Service` exactly one `SwComponentPrototype` typed by a `ServiceSwComponentType` is created based on the `ServiceSwComponentType`. Additionally, the connectors are constructed that connect the pairs of `PortPrototypes` belonging to the `SwComponentPrototypes` requiring services and those belonging to the actual services. `]()`
- **[TPS_SWCT_01407] Creation of `InternalBehavior` typed by a `ServiceSwComponentType`** `[` For each `ServiceSwComponentType` an `SwcInternalBehavior` is created or extended providing the information about `PortDefinedArgumentValues`, `RunnableEntitys` and `RTEEvents` necessary for RTE generation. `]()`

Further detailing of the service ports by filling in these `PortDefinedArgumentValues` is also done in ECU Configuration phase. See also chapter 7.6.3.

- **[TPS_SWCT_01408] Creation of `SwcBswMapping`** `[` For the RTE module configuration an implementation of the AUTOSAR `Service` described by a `Basic Software Module Description` needs to be selected. The `SwcBswMapping` to the corresponding `SwComponentPrototype` needs to be created accordingly.

For each `SwcInternalBehavior` one `SwcImplementation` is being created. The information for `SwcImplementation` should be generated based on the available information of `BswImplementation`¹. `]()`

- **[TPS_SWCT_01409] Update of `PortDefinedArgumentValues`** `[` Depending of the configuration of the `Service BSW` it might be necessary to update the `ValueSpecifications` belonging to the `PortDefinedArgumentValues` generated in a previous step. `]()`

Class	ServiceNeeds (abstract)			
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
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			
Attribute	Datatype	Mul.	Kind	Note
–	–	–	–	–

Table 11.2: ServiceNeeds

¹This step does in general not require copying any attributes or elements aggregated in `BswImplementation` into the generated instance of `SwcImplementation` since the only mandatory information for the RTE configuration is the reference from `SwcImplementation` to the selected `SwcInternalBehavior`.

11.2 Extending the ECU Software Composition

As explained in chapter 11.1, `Service Configuration` takes place in ECU Configuration phase. In the ECU extract of the `System`, the Software Components and their ECU-internal connectors are represented as a flat set aggregated by `RootSwCompositionPrototype` as indicated in Figure 11.1.

ECU Configuration extends this aggregation by adding `SwComponentPrototypes` (each typed by a specific `ServiceSwComponentType`) and the required `AssemblySwConnectors` to the `RootSwCompositionPrototype`. This is possible without changing the initial artifacts of the ECU extract, because these aggregations are stereotyped as `«atpSplittable»` in the meta-model.

After this step, the `RootSwCompositionPrototype` (denoted by `EcucValueCollection.ecuExtract.rootSoftwareComposition`) represents the whole Software Composition on the given ECU. This collection includes both the software components mapped to the ECU **and** the necessary service components represented as one `SwComponentPrototype` for each AUTOSAR `Service` utilized on the given ECU.

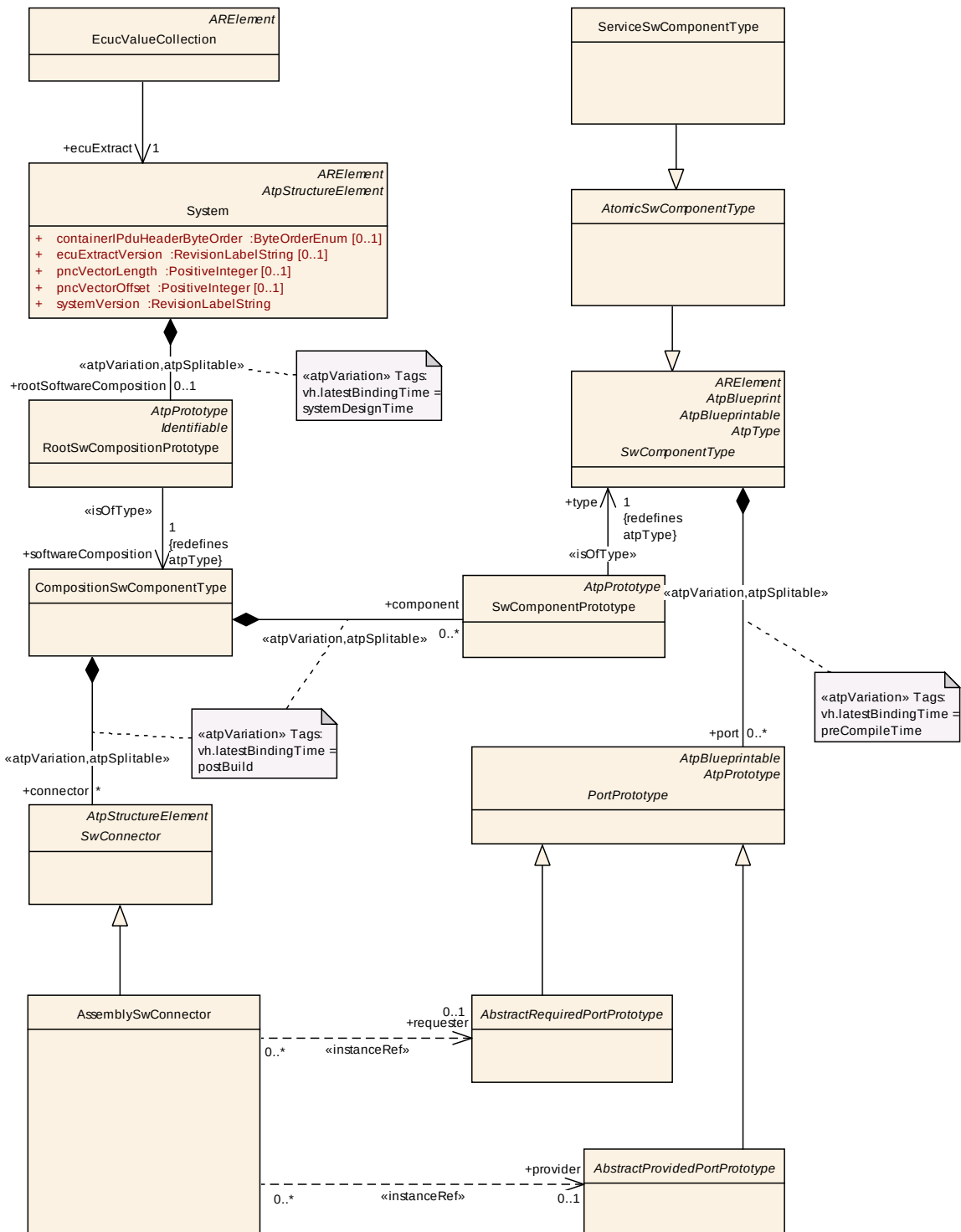


Figure 11.1: Usage of **RootSwCompositionPrototype** on an ECU

11.3 Service Software Component Type

As mentioned in [TPS_SWCT_01405], AUTOSAR Services are represented by a meta model class of their own, the **ServiceSwComponentType**. As can be seen

in Figure 11.2 `ServiceSwComponentType` is a specialization of `AtomicSwComponentType`.

Like any other `SwComponentType` they can aggregate `PortPrototypes`.

[constr_2019] `ServiceSwComponentType` shall have service ports only [In the case of `ServiceSwComponentType`, all aggregated `PortPrototypes` need to have an `<<isOfType>>` relationship to a `PortInterface` which has its `isService` attribute set to `true`. The exceptions described in [TPS_SWCT_01572], [TPS_SWCT_01579] and [TPS_SWCT_01580] apply.]()

[TPS_SWCT_01579] `Dcm` can directly access `dataElements` in `PPortPrototypes` typed by a `SenderReceiverInterface` [An exception from the rule described in [constr_2019] applies: the `Dcm` can directly access `dataElements` in `PortPrototypes` (that is both `AbstractProvidedPortPrototype` and `AbstractRequiredPortPrototype`) typed by a `SenderReceiverInterface`.

For this purpose, the `ServiceSwComponentType` that represents the `Dcm` functionality can have `AbstractProvidedPortPrototypes` and `AbstractRequiredPortPrototypes` typed by a compatible `SenderReceiverInterface` that may set `isService` to `FALSE`.]()

[TPS_SWCT_01580] `Dem` can directly access `dataElements` in `PPortPrototypes` typed by a `SenderReceiverInterface` [An exception from the rule described in [constr_2019] applies: the `Dem` can directly access `dataElements` in `AbstractProvidedPortPrototypes` typed by a `SenderReceiverInterface`.

For this purpose, the `ServiceSwComponentType` that represents the `Dem` functionality can have `RPortPrototypes` typed by a compatible `SenderReceiverInterface` that may set `isService` to `FALSE`.]()

[TPS_SWCT_01411] Use cases for a `ServiceSwComponentType` to express `ServiceNeeds` [There are valid use cases for a `ServiceSwComponentType` to express `ServiceNeeds`². This leads to a situation where `ServiceSwComponentTypes` are iteratively created in response to `ServiceNeeds` expressed by other `ServiceSwComponentTypes`. Please refer to the AUTOSAR methodology [4] for more details about how this shall be implemented into the workflow.]()

Similar to an `EcuAbstractionSwComponentType` and a `ComplexDeviceDriverSwComponentType`, the `ServiceSwComponentType` represents a hybrid concept between Software Component and Basic Software Module. The BSW part is described by the means of the BSW Module Description Template [7].

The mapping between the `ServiceSwComponentType` and the corresponding `BswModuleDescription` is provided by the class `SwcBswMapping` which in addition also maps the two corresponding `InternalBehaviors` (see [TPS_SWCT_01408]. This mechanism is further explained in [7].

²Thereby the previously existing constraint 1127 becomes invalid.

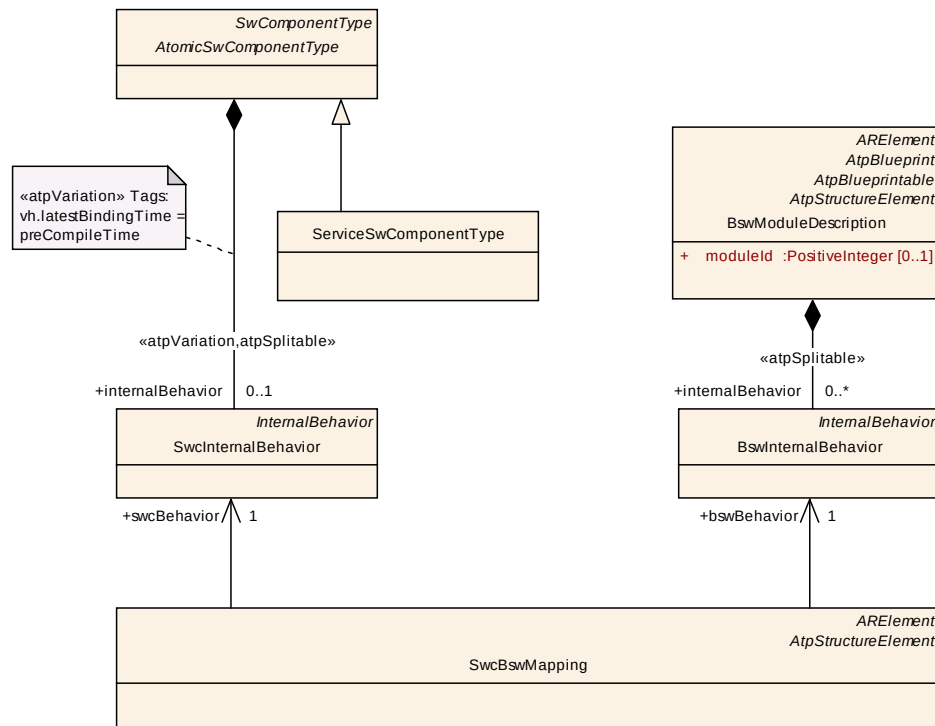


Figure 11.2: **ServiceSwComponentType**

Class	ServiceSwComponentType			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
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, ARObjct, AtomicSwComponentType, AtpBlueprint, AtpBlueprintable, Atp Classifier, AtpType, CollectableElement, Identifiable, Multilanguage Referrable, PackageableElement, Referrable, SwComponentType			
Attribute	Datatype	Mul.	Kind	Note
—	—	—	—	—

Table 11.3: **ServiceSwComponentType**

[TPS_SWCT_01412] **ServiceSwComponentType** shall be added in ECU Configuration phase [**ServiceSwComponentType** shall not be used when modeling application software using **CompositionSwComponentType**; they are only added in ECU Configuration phase where exactly one **SwComponentPrototype** per **ServiceSwComponentType** per ECU is added to the ECU Description model.

The Base ECU Config Generator tool needs to take care that for all service ports of **SwComponentPrototypes** mapped to the ECU service ports at the appropriate **ServiceSwComponentTypes** are created. In the process the specified communication pattern A, B, or C for a specific kind of service port has to be considered, see table 11.1.

In case of pattern A for each different type of service port one port on the `ServiceSwComponentType` is created.

In case of pattern B and C for each service port of a `SwComponentPrototype` one port on the `ServiceSwComponentType` is created.

More explicitly, all instances of `AtomicSwComponentType` need to be checked for `PortPrototypes` of `PortInterfaces` with `isService` attribute set to `true` and referenced by `ServiceNeeds` and for each of these `PortInterface` instances belonging to the AUTOSAR `Service` to be configured one `PortPrototype` implementing the same or a compatible `PortInterface` needs to be created on the `ServiceSwComponentType`. `]()`

[TPS_SWCT_02500] Roles on Application/Service Components need to Match `[` The roles of the `PortPrototypes` (required/provided) on the Application Component and the Service Component side obviously need to match. For example an `RPortPrototype` attached to an application `AtomicSwComponentType` matches a `PPortPrototype` attached to a `ServiceSwComponentType`. `]()`

11.4 Service Proxy Component Type

[TPS_SWCT_01413] Local communication with services `[` Application software components may communicate with an instance of a `ServiceSwComponentType` only locally on an ECU. `]()`

[TPS_SWCT_01414] Mode manager needs to communicate with application software components located on other ECUs `[` There are however use cases for the application and vehicle mode management, where a mode manager (namely the `Basic Software Mode Manager`, see [15]) is part of the basic software but conceptually still needs to communicate with application software components located on other ECUs (as exemplified by Figure 11.3).

In order to make this communication possible, the `ServiceProxySwComponentType` is used.

For the application software and the RTE it behaves like a “normal” `AtomicSwComponentType`, but it is actually a proxy for an AUTOSAR `Service`. `]()`

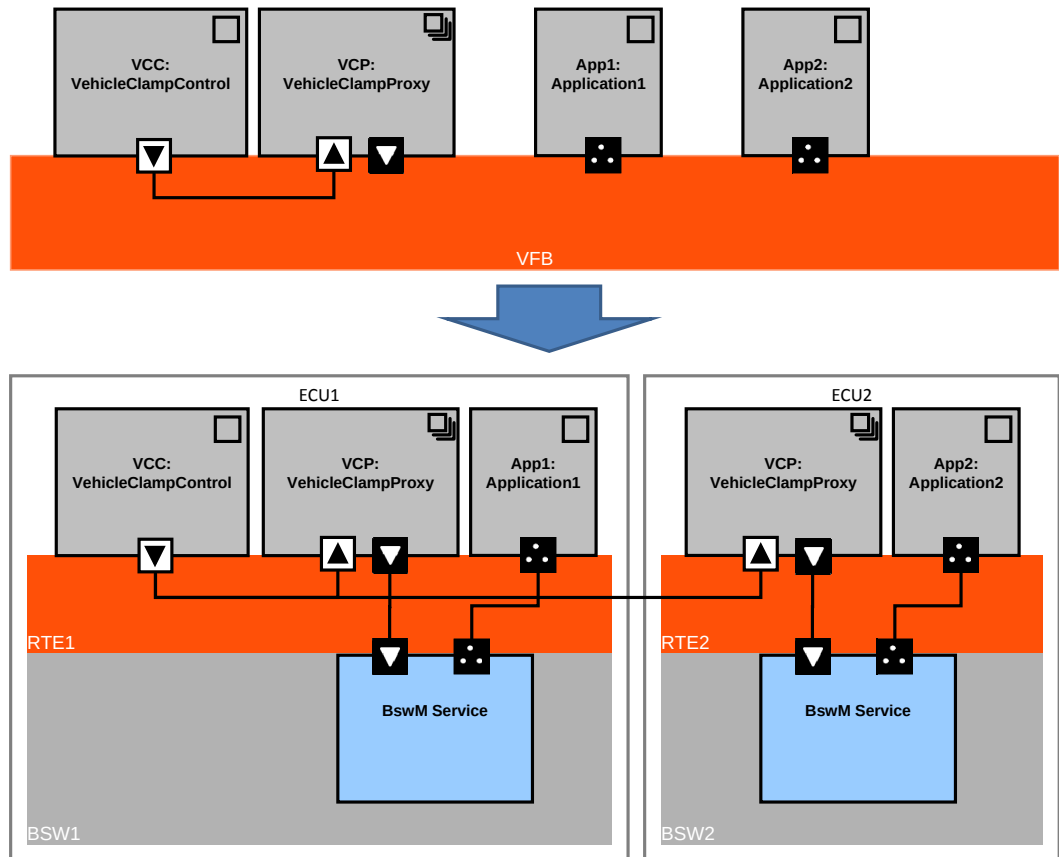


Figure 11.3: Mode request over the network [3]

[TPS_SWCT_01415] Interfaces of **ServiceProxySwComponentType** [This means that on the one side it has to communicate over service ports with the ECU-local **ServiceSwComponentType** it represents. On the other side it has to offer the corresponding **PortPrototypes** to the **ApplicationSwComponentTypes**.]()

In the meta-model, the **ServiceProxySwComponentType** does not differ from an **ApplicationSwComponentType** except by its class. It is up to the implementer to meet the restrictions imposed by the semantics as a proxy.

[TPS_SWCT_01416] Difference between a **ServiceProxySwComponentType** and an **ApplicationSwComponentType** [The main difference between a **ServiceProxySwComponentType** and an **ApplicationSwComponentType** is on system level:

A prototype of a **ServiceProxySwComponentType** can be mapped to several ECUs even if it appears only once in the VFB system, because such a prototype is required on each ECU, where it has to address a local **ServiceSwComponentType**.

As a result of this, a **ServiceProxySwComponentType** can only receive but not send signals over the network. More details are explained in the class table below.]()

Class	ServiceProxySwComponentType			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	This class provides the ability to express a software-component which provides access to an internal service for remote ECUs. It acts as a proxy for the service providing access to the service. An important use case is the request of vehicle mode switches: Such requests can be communicated via sender-receiver interfaces across ECU boundaries, but the mode manager being responsible to perform the mode switches is an AUTOSAR Service which is located in the Basic Software and is not visible in the VFB view. To handle this situation, a ServiceProxySwComponentType will act as proxy for the mode manager. It will have R-Ports to be connected with the mode requestors on VFB level and Service-Ports to be connected with the local mode manager at ECU integration time. Apart from the semantics, a ServiceProxySwComponentType has these specific properties: • A prototype of it can be mapped to more than one ECUs in the system description. • Exactly one additional instance of it will be created in the ECU-Extract per ECU to which the prototype has been mapped. • For remote communication, it can have only R-Ports with sender-receiver interfaces and 1:n semantics. • There shall be no connectors between two prototypes of any ServiceProxySwComponentType. Tags: atp.recommendedPackage=SwComponentTypes			
Base	ARElement , ARObject , AtomicSwComponentType , AtpBlueprint , AtpBlueprintable , AtpClassifier , AtpType , CollectableElement , Identifiable , MultilanguageReferrable , PackageableElement , Referrable , SwComponentType			
Attribute	Datatype	Mul.	Kind	Note
—	—	—	—	—

Table 11.4: ServiceProxySwComponentType

[constr_2016] Connections between [SwComponentPrototypes](#) of type [ServiceProxySwComponentType](#) [A connection between [PortPrototypes](#) belonging to [SwComponentPrototypes](#) where both are typed by [ServiceProxySwComponentType](#) is not permitted.]()

[constr_2017] Ports of [ServiceProxySwComponentTypes](#) [[ServiceProxySwComponentType](#) is only permitted to define

- [RPortPrototypes](#) that are typed by [SenderReceiverInterface](#) or
- [PortPrototypes](#) that are typed by a [PortInterface](#) where the [isService](#) attribute is set to true.

]()

[constr_2018] Supported remote communication of a [ServiceProxySwComponentType](#) [For remote communication, [ServiceProxySwComponentType](#) can have only [RPortPrototypes](#) typed by [SenderReceiverInterfaces](#) in a 1:n communication scenario.]()

11.5 Non Volatile Memory

11.5.1 Introduction

The AUTOSAR Architecture defines two alternatives how a software component can access non volatile memory.

- The first option is that the software component defines in its [InternalBehavior](#) a [PerInstanceMemory](#) and a [NvBlockNeeds](#) referring to the [PerInstanceMemory](#) via a [RoleBasedDataAssignment](#).

In this case the `NVRAM Block` is exclusively accessed by this software component and the *NvM* [31]. Therefore the *nv data* is encapsulated inside the software component and can not be accessed directly by other software components.

The [PerInstanceMemory](#) can be typed with [AutosarDataTypes](#) in the case of [arTypedPerInstanceMemory](#) or with C data types in the case of [perInstanceMemory](#). For further information see section 7.7 and 7.11.3.

- The second option is that the software component uses communication based on [PortPrototypes](#) to access *nv data* provided by a [NvBlockSwComponentType](#).

In this case it is possible that *nv data* used by different [AtomicSwComponentTypes](#) is packed in one larger `NVRAM Block` to reduce the `NVRAM Block` management overhead or that the same *nv data* used by several software components with a reduced RAM overhead. The *nv data* of a [NvBlockSwComponentType](#) is typed with [AutosarDataTypes](#).

More details regarding particular scenarios of interacting with the *NvM* [31] can be found in section 7.11.3.1.

11.5.2 NvBlockComponent

[TPS_SWCT_01142] non-volatile data are provided by a specialized [AtomicSwComponentType](#) [On the VFB [3], the non-volatile data are provided by a specialized [AtomicSwComponentType](#), the [NvBlockSwComponentType](#).

An [NvBlockSwComponentType](#) can represent one or more `NVRAM Blocks` managed by the *NVRAM Manager*. The *nv data* [PortPrototypes](#) of the [NvBlockSwComponentType](#) are exclusively typed by [NvDataInterfaces](#).]([RS_SWCT_03225](#))

[TPS_SWCT_01143] Non-volatile data represented by an `NvBlockSwComponentType` can be read and written [The non-volatile data represented by an `NvBlockSwComponentType` can be read and written. For this purpose the `NvBlockSwComponentType` is allowed to have `PPortPrototypes` and `RPortPrototypes`.]([RS_SWCT_03225](#))

Additionally, the `NvBlockSwComponentType` might have client server `PortPrototypes` to offer the block-related services, administrative services or notifications.

[constr_2009] Supported kinds of `PortPrototypes` of a `NvBlockSwComponentType` [With respect to external communication, `NvBlockSwComponentType` is limited to the definition of the following kinds of `PortPrototype`:

- `PortPrototypes` typed by either `NvDataInterfaces` or `ClientServerInterfaces`
- `RPortPrototypes` typed by `ModeSwitchInterfaces`

]()

[constr_2010] Connections between `SwComponentPrototypes` of type `NvBlockSwComponentType` [The existence of `SwConnectors` that refer to `PortPrototypes` belonging to `SwComponentPrototypes` where both are typed by `NvBlockSwComponentType` is not permitted.]()

Class	NvBlockSwComponentType			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	The <code>NvBlockSwComponentType</code> defines non volatile data which data can be shared between <code>SwComponentPrototypes</code> . The non volatile data of the <code>NvBlockSwComponentType</code> are accessible via provided and required ports. Tags: atp.recommendedPackage=SwComponentTypes			
Base	ARElement , ARObject , AtomicSwComponentType , AtpBlueprint , AtpBlueprintable , AtpClassifier , AtpType , CollectableElement , Identifiable , MultilanguageReferrable , PackageableElement , Referrable , SwComponentType			
Attribute	Datatype	Mul.	Kind	Note
nvBlockDescriptor	NvBlockDescriptor	*	aggr	Specification of the properties of exactly one NVRAM Block. Stereotypes: atpSplittable; atpVariation Tags: atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime

Table 11.5: NvBlockSwComponentType

Class	NvDataInterface			
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
Note	A non volatile data interface declares a number of VariableDataPrototypes to be exchanged between non volatile block components and atomic software components. Tags: atp.recommendedPackage=PortInterfaces			
Base	ARElement, ARObjct, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, CollectableElement, DataInterface, Identifiable, Multilanguage Referrable, PackageableElement, PortInterface, Referrable			
Attribute	Datatype	Mul.	Kind	Note
nvData	VariableDataPrototype	1..*	aggr	The VariableDataPrototype of this nv data interface.

Table 11.6: NvDataInterface

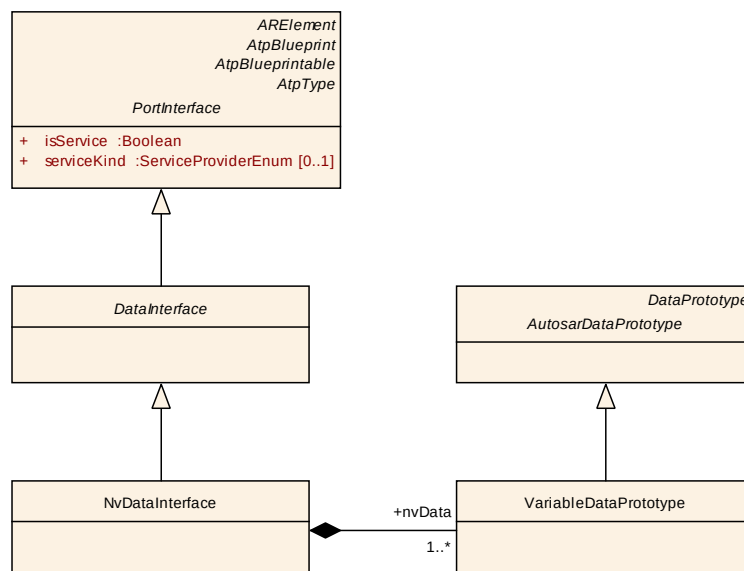


Figure 11.4: NvDataInterface

11.5.3 Software-Components using *NVRAM data* of NvBlockComponents

[constr_2011] **Connections between SwComponentPrototypes typed by NvBlockSwComponentType and SwComponentPrototypes typed by other AtomicSwComponentTypes** [The *nv data PortPrototypes* of the *SwComponentPrototype* typed by an *NvBlockSwComponentType* are either connected with *PortPrototypes* typed by *NvDataInterfaces* or *SenderReceiverInterfaces* of other *AtomicSwComponentType*.]()

[constr_1148] **PortInterfaces of PortPrototypes used to connect to NvBlockSwComponentTypes** [*PortInterfaces* of *PortPrototypes* used to connect to *NvBlockSwComponentTypes* as well as the *PortInterfaces* used in the context of *NvBlockSwComponentTypes* shall **always** set the value of the attribute *isService* to *false*.]()

[constr_1149] **PortPrototypes** used for NV data management [A **PortPrototype** typed by a **ClientServerInterface** used for NV data management, i.e. the interaction of **ApplicationSwComponentTypes** with **NvBlockSwComponentTypes**, shall be typed by **ClientServerInterfaces** that are compatible to the particular **ClientServerInterfaces** derived from MOD_GeneralBlueprints [30]. [constr_1148] applies.]()

For details see chapter 6.4.4.

Note: In case of *nv data* which is read and written and shared between several **SwComponentPrototypes** the **NvBlockSwComponentType** establishes a not directly obvious kind of communication. Nevertheless this is intentionally supported and it is under responsibility of the VFB designer to take care that only *nv data* is shared where the functionality of the software components is not impaired.

To determine for an VFB designer which *nv data* can be potentially by mapped into the same NVRAM Block a software-component can specify further attributes for its *nv data PortPrototypes* by the definition of **SwcServiceDependency(s)** with **NvBlockNeeds**. In this case the role attribute of the **assignedPort** has to be set to the value **NvDataPort**. This aspect is also explained in section 7.11.3.1.4.

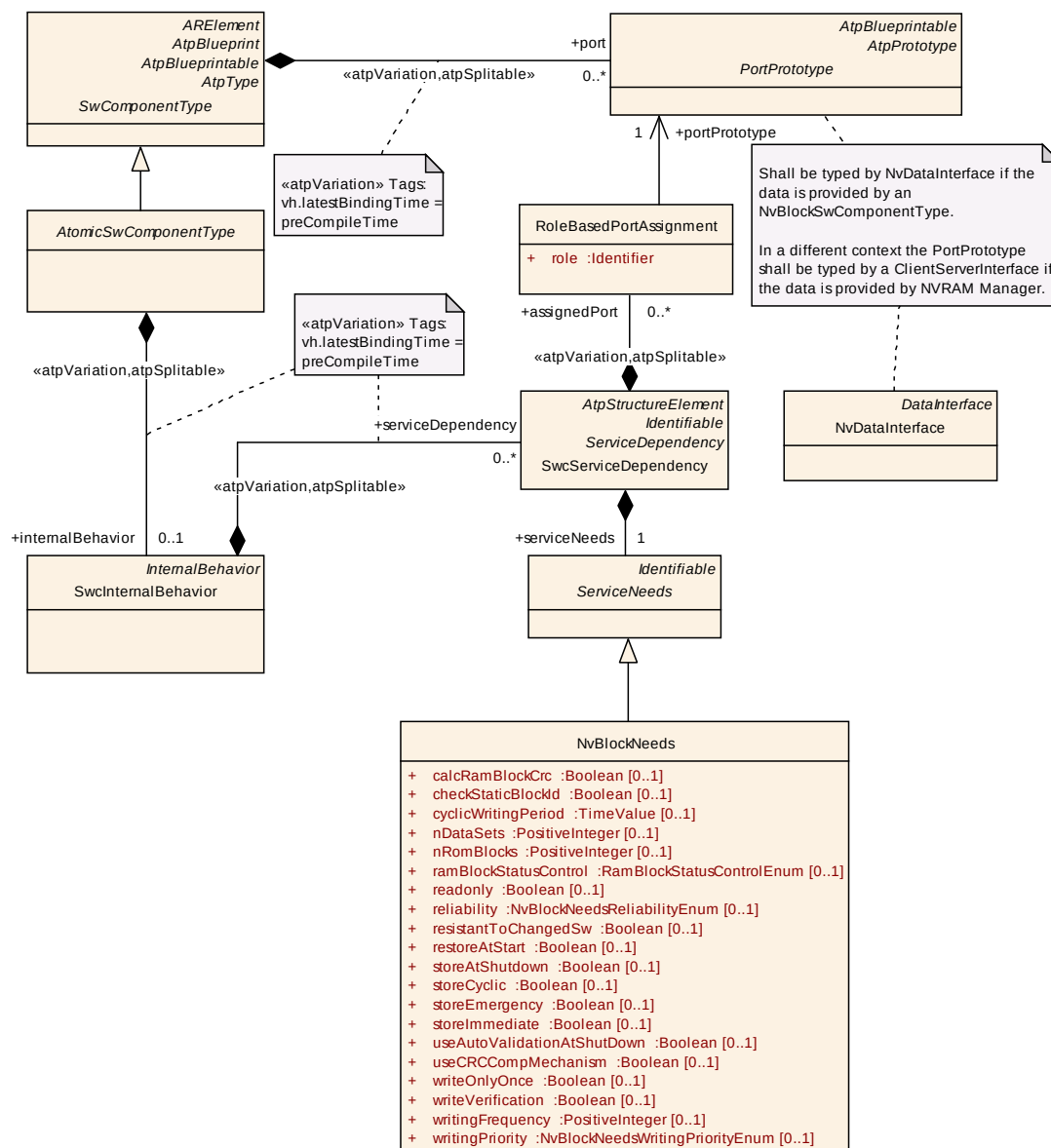


Figure 11.5: NvBlockNeeds for *nv data* PortPrototypes

In contrast to the `NvBlockNeeds` that describe the expected configuration of a whole NVRAM Block, the `NvBlockNeeds` for `nv data PortPrototypes` defines only the attributes which are required from the point of view of a software-component to ensure its functionality.

This means an empty attribute has the semantic of “don’t care”.

Further on the VFB designer has got the freedom to specify how the requested `NvRAMBlock` attributes are fulfilled by the created `NvBlockDescriptor`.

For instance, *nv data* with different `writingFrequency` might be mapped to one NVRAM Block. In this case the `NvBlockNeeds` of the `NvBlockDescriptor` has to indicate the worst case which is the higher frequency.

The recommended relationship is shown in table 11.7. But please note that this table does not represent a binding constraint.

Attribute of NvBlockNeeds	NvBlockNeeds of different <i>nv data PortPrototypes</i> of software-components	NvBlockNeeds of NvBlockDescriptor
readonly	Recommended to match for all connected <i>nv data PortPrototypes</i> if specified.	Recommended to be identical as requested by <i>nv data PortPrototypes</i> .
reliability	Can be different.	Recommended to be set to the highest reliability class request by any mapped <i>nv data PortPrototypes</i> .
resistantToChangedSw	Recommended to match for all connected <i>nv data PortPrototypes</i> if specified.	Recommended to be identical as requested by <i>nv data PortPrototypes</i> .
restoreAtStart	Recommended to match for all connected <i>nv data PortPrototypes</i> if specified.	Recommended to be identical as requested by <i>nv data PortPrototypes</i> .
storeAtShutdown	Recommended to match for all connected <i>nv data</i> ports if specified.	Recommended to set to true if any of the <i>nv data PortPrototypes</i> requests writing at shutdown.
writeOnlyOnce	Recommended to match for all connected <i>nv data PortPrototypes</i> if specified.	Recommended to be identical as requested by <i>nv data PortPrototypes</i> .
writingFrequency	Can be different.	Recommended to be set to the highest requested frequency of the mapped <i>nv data PortPrototypes</i> .
writingPriority	Can be different.	Recommended to be set to the highest requested frequency of the mapped <i>nv data PortPrototypes</i> .
writeVerification	Can be different.	Recommended to set to true if any of the <i>nv data PortPrototypes</i> requests a write verification.
calcRamBlockCrc	Can be different.	Recommended to set to true if any of the <i>nv data PortPrototypes</i> requests a CRC calculation.
checkStaticBlockId	Can be different.	Recommended to set to true if any of the <i>nv data PortPrototypes</i> requests a check of the static block ID.
ramBlockStatusControl	Can be different.	Recommended to set to true if any of the <i>nv data PortPrototypes</i> requests a use of the API for accessing the block.
storeAtShutdown	Can be different.	Recommended to set to true if any of the <i>nv data PortPrototypes</i> requests writing at shutdown.
storeCyclic	Can be different.	Recommended to set to true if any of the <i>nv data PortPrototypes</i> requests cyclic writing.
storeEmergency	Can be different.	Recommended to set to true if any of the <i>nv data PortPrototypes</i> requests emergency writing.
storeImmediate	Can be different.	Recommended to set to true if any of the <i>nv data PortPrototypes</i> requests immediate writing.

Table 11.7: NvBlockNeeds dependencies

With respect to the completeness of table 11.7 (which intentionally doesn't contain a remark regarding the value of `cyclicWritingPeriod`), it should be noted that (according to [TPS_SWCT_01585]) the value of `NvBlockDescriptor.nvBlockNeeds.cyclicWritingPeriod` shall be ignored in favor of `NvBlockDescriptor.timingEvent.period`.

Therefore, the missing statement for `cyclicWritingPeriod` in the spirit of table 11.7 is that the values of `SwcServiceDependency.serviceNeeds.cyclicWritingPeriod` can be different from the value of `NvBlockDescriptor.timingEvent.period`.

It is recommended that the value of `NvBlockDescriptor.timingEvent.period` shall be set to the lowest requested time value of the mapped *nv data PortPrototypes* (implemented by `SwcServiceDependency.serviceNeeds.cyclicWritingPeriod`).

11.5.4 NvBlockDescriptor

[TPS_SWCT_01144] **NvBlockDescriptor** specifies the properties of exactly one **NVRAM Block** [A `NvBlockDescriptor` specifies the properties of exactly one **NVRAM Block** of a `NvBlockSwComponentType`.

It contains information about the requested **NVRAM Block** configuration of the *NVRAM Manager*, `ramBlock` and `romBlock`, the mapping between the `PortPrototypes` of the `NvBlockSwComponentType` and the data inside a `ramBlock` as well as the role of the `clientServerPorts` expressed in terms of `RoleBasedPortAssignment`.
]()

Class	NvBlockDescriptor			
Package	M2::AUTOSARTemplates::SWComponentTemplate::NvBlockComponent			
Note	Specifies the properties of exactly on NVRAM Block.			
Base	ARObject, AtpClassifier, AtpFeature, AtpStructureElement, Identifiable , Multilanguage Referrable, Referrable			
Attribute	Datatype	Mul.	Kind	Note
clientServerPort	RoleBasedPortAssignment	*	aggr	The <code>RoleBasedPortAssignment</code> defines which client server port of the <code>NvBlockSwComponentType</code> serves for which kind of service or notification. In case of notifications one common callback function is provided by the RTE for each individual kind of notification defined by the "role". The aggregation of <code>RoleBasedPortAssignment</code> is subject to variability with the purpose to support the conditional existence of ports. Stereotypes: <code>atpVariation</code> Tags: <code>vh.latestBindingTime=preCompileTime</code>

Attribute	Datatype	Mul.	Kind	Note
constantValueMapping	ConstantSpecificationMappingSet	*	ref	Reference to the ConstantSpecificationMapping to be applied for the particular NVRAM Block Stereotypes: atpSplitable Tags: atp.Splitkey=constantValueMapping
dataTypeMapping	DataTypeMappingSet	*	ref	Reference to the DataTypeMapping to be applied for the particular NVRAM Block. Stereotypes: atpSplitable Tags: atp.Splitkey=dataTypeMapping
instantiationDataDefProps	InstantiationDataDefProps	*	aggr	The purpose of InstantiationDataDefProps are the refinement of some data def properties of individual instantiations within the context of a NvBlockSwComponentType. The aggregation of InstantiationDataDefProps is subject to variability with the purpose to support the conditional existence of ports, component internal memory objects and those attributes. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime
nvBlockDataMapping	NvBlockDataMapping	1..*	aggr	Defines the mapping between the VariableDataPrototypes in the NvBlockComponents ports and the VariableDataPrototypes of the RAM Block. The aggregation of NvBlockDataMapping is subject to variability with the purpose to support the conditional existence of nv data ports. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime
nvBlockNeeds	NvBlockNeeds	1	aggr	Specifies the abstract needs on the configuration of the NVRAM Manager for the single NVRAM Block described by this NvBlockDescriptor. In addition, it may define requirements for writing strategies in an implementation of an NvBlockSwComponentType by the RTE. Please note that the attributes nDataSets and nRomBlocks are not relevant for this aggregation because the RTE will allocate just one block anyway. In a different context, however, they do make sense.
ramBlock	VariableDataPrototype	1	aggr	Defines the RAM Block of the NVRAM Block provided by NvBlockSwComponentType.
romBlock	ParameterDataPrototype	0..1	aggr	Defines the ROM Block of the NVRAM Block provided by NvBlockSwComponentType.

Attribute	Datatype	Mul.	Kind	Note
supportDirtyFlag	Boolean	0..1	attr	Specifies whether calling of NvM functions for writing and/or status control of potentially modified RAM Blocks to NV memory shall be controlled by the RTE.
timingEvent	TimingEvent	0..1	ref	this reference can be taken to identify the TimingEvent to be used by the RTE for implementing a cyclic writing strategy for this block

Table 11.8: NvBlockDescriptor

For more explanation about the semantics of the attribute `NvBlockDescriptor.supportDirtyFlag` please refer to the SWS RTE [2].

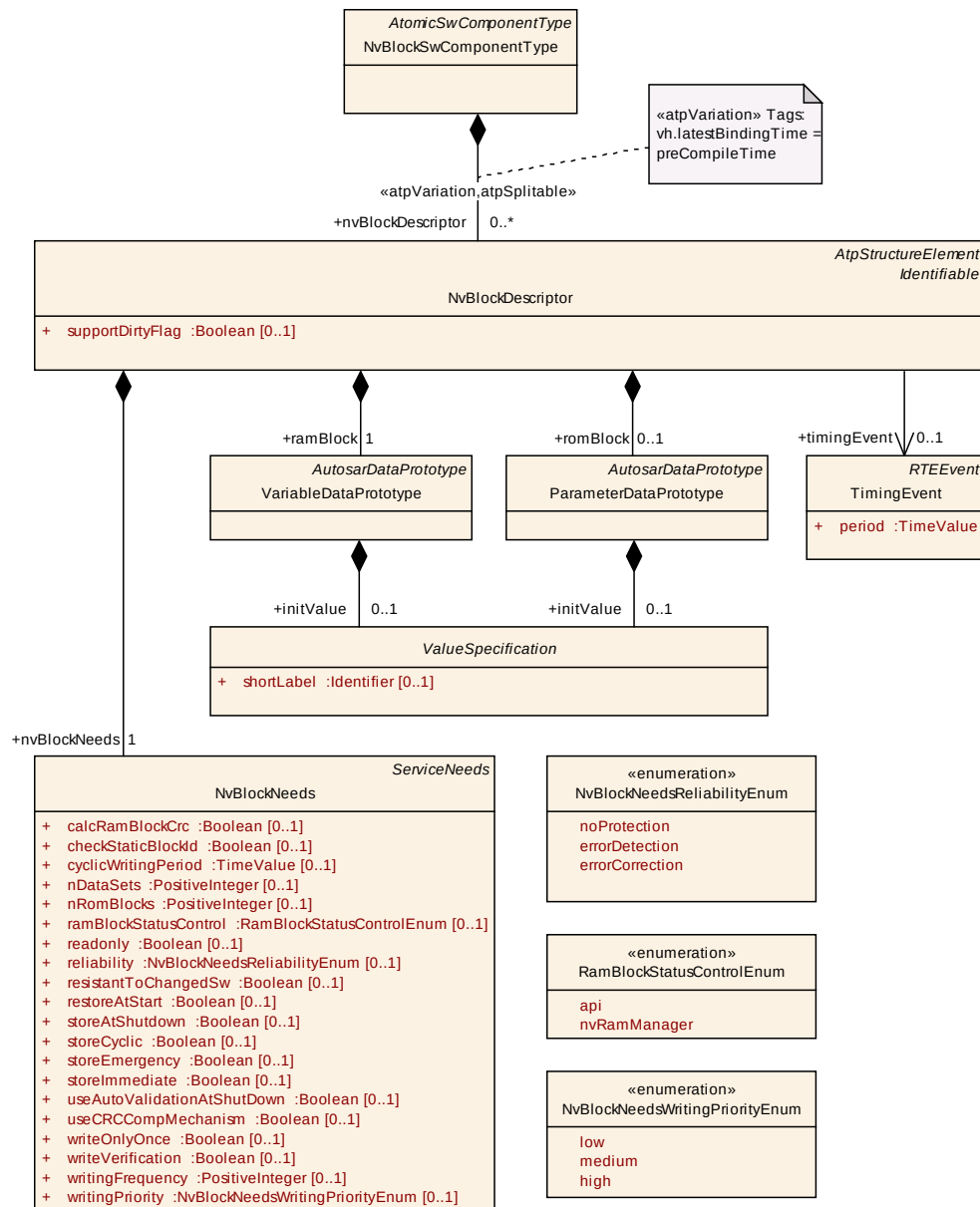


Figure 11.6: NvBlockSwComponentType and NvBlockDescriptor

[constr_1095] Values of `nDataSets` vs. `reliability` [If the value of `nDataSets` is greater than 0 the value of `reliability` shall not be set to `errorCorrection`.]()

The reason for the existence of [constr_1095] is that the AUTOSAR NvM [31] does not support error correction for NV data sets.

If the value of `nDataSets` is equal to 0 the value of `reliability` can take any value out of `NvBlockNeedsReliabilityEnum`.

11.5.4.1 Writing Strategies

[TPS_SWCT_01586] Writing strategies for *nv data* [By setting certain attributes in the meta-class `NvBlockDescriptor` it is possible to configure different writing strategies for the values of an RAM Block to the NVRAM storage. [constr_1310] applies.

The following use cases are supported:

- Write data **cyclically**. This use case requires the existence of attribute `NvBlockDescriptor.nvBlockNeeds.storeCyclic` with the value `true` and also attribute `NvBlockDescriptor.cyclicWritingPeriod` needs to exist and have a reasonable value.

In the context of using the attribute `NvBlockDescriptor.cyclicWritingPeriod` the constraints [constr_1308] and [constr_1309] apply.

Please refer to [TPS_SWCT_01587] and Figure 11.7 for more information about how this aspect can be configured.

- Write data **immediately**. This means that data send to the `NvBlockSwComponentType` will be written immediately to NVRAM storage.

This use case corresponds to setting the value of attribute `NvBlockDescriptor.nvBlockNeeds.storeImmediate` to the value `true`.

Please refer to [TPS_SWCT_01588] and Figure 11.8 for more information about how this aspect can be configured.

- Write on **emergency**. With this setting, data shall be written to NVRAM storage if the ECU fails in some way.

This use case corresponds to setting the value of attribute `NvBlockDescriptor.nvBlockNeeds.storeEmergency` to `true`.

As explained in [TPS_SWCT_01589], setting the value of this attribute is not sufficient to achieve the intended semantics.

- Write at **shutdown**. Here, the data are written to NVRAM storage when the ECU shuts down.

This use case corresponds to setting the value of attribute `NvBlockDescriptor.nvBlockNeeds.storeAtShutdown` to `true`.

|(RS_SWCT_03225)

Of course, the actual implementation of the different writing strategies goes beyond setting the value of attributes and requires the existence of dedicated `RunnableEntities` in the `SwcInternalBehavior` of the enclosing `NvBlockSwComponentType` that are triggered in response to `RTEEvents` applicable for the particular use case.

[TPS_SWCT_01587] The cyclic writing of *nv data* requires the existence of a `TimingEvent` | The implementation of cyclic writing of *nv data* requires the existence of a `TimingEvent` that can be taken to trigger a corresponding `RunnableEntity` that in turn takes care of calling the respective APIs for writing the data.
|(RS_SWCT_03225)

This aspect is depicted in Figure 11.7.

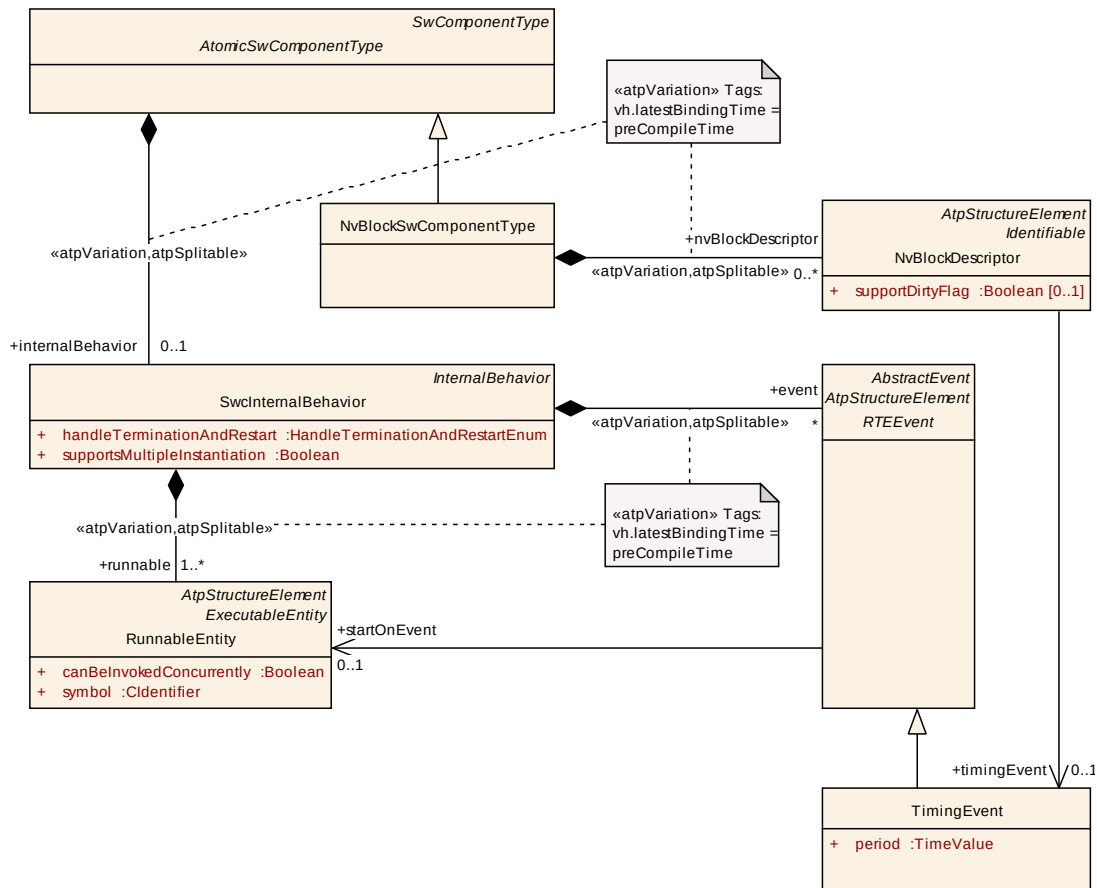


Figure 11.7: How to model a cyclic writing strategy for *nv data*

[TPS_SWCT_01588] `DataReceivedEvent` for storing *nv data* immediately | The approach to store data immediately after reception by an `NvBlockSwComponentType` requires the activation of a `RunnableEntity` by a `DataReceivedEvent`.
|(RS_SWCT_03225)

This approach is depicted in Figure 11.8.

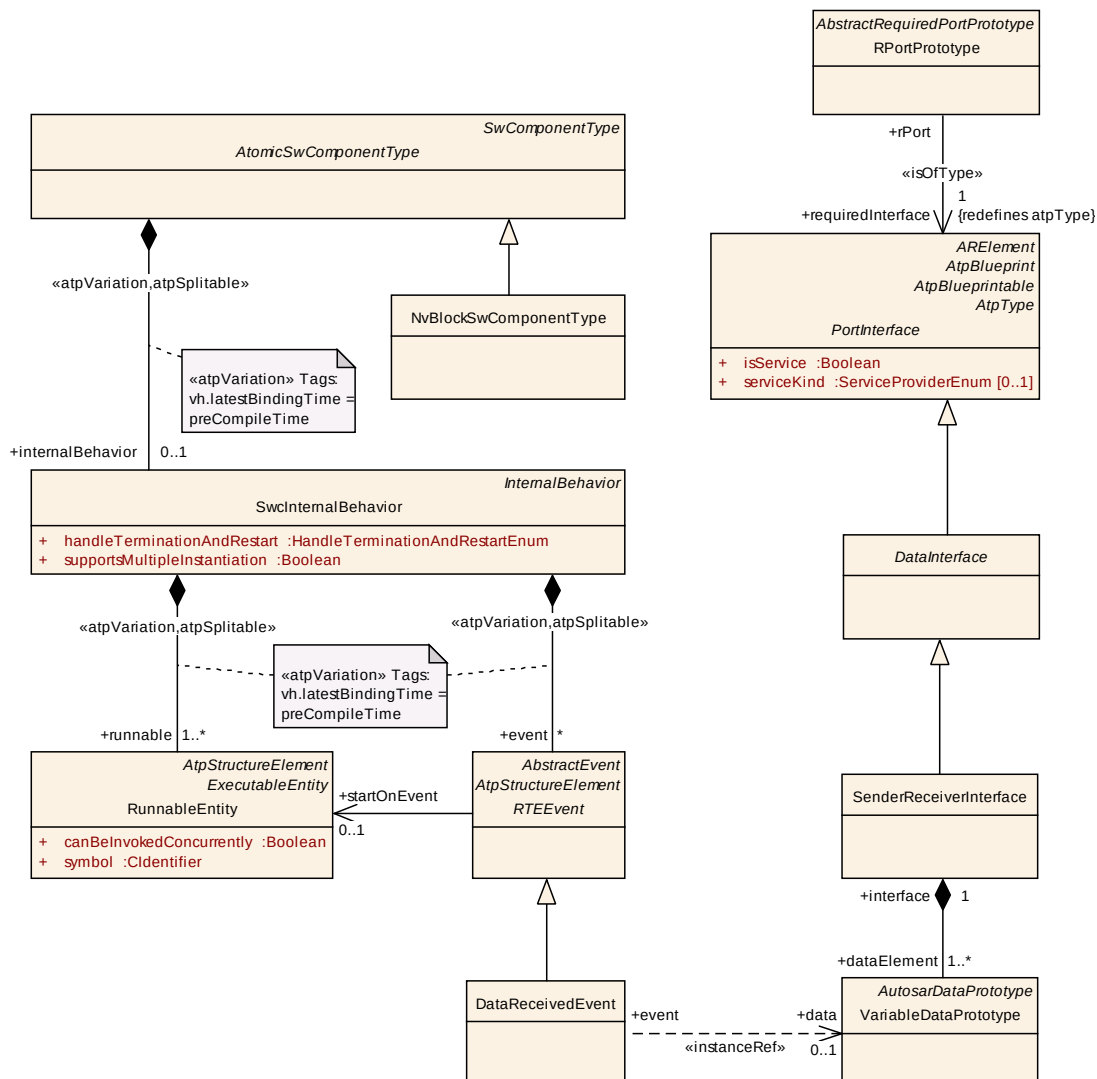


Figure 11.8: How to model an immediate writing strategy for *nv data*

[TPS_SWCT_01589] Implementation of emergency storing of *nv data* [The use case for [storeEmergency](#) can only be implemented by means of a Complex Driver.

In particular, the `Complex Driver` is responsible for the detection of an ECU failure. If a relevant error occurs the `Complex Driver` should call the `NvM` write block operation for the emergency blocks directly.]([RS_SWCT_03225](#))

This consequently means that the `NvM` shall react to write operations coming from the `Complex Driver` by giving them the highest priority (re-queuing of `NvM` write block requests).

Please note that the behavior described in [\[TPS_SWCT_01587\]](#) in general is supported by AUTOSAR by requiring that `NVRAM Blocks` shall have to be configured

with “immediate priority”. The technical implications are explained in the respective SWS [31], e.g. in [SWS_NvM_00182] and [SWS_NvM_00300].

[TPS_SWCT_01590] Combination of writing strategies for *nv data* is possible [AUTOSAR positively supports the configuration of a combination of writing strategies for *nv data*.]([RS_SWCT_03225](#))

In other words, in consequence of [\[TPS_SWCT_01590\]](#) it is possible that (for example) both [NvBlockDescriptor.storeImmediate](#) as well as [NvBlockDescriptor.storeCyclic](#) may exist and set to `true` in the context of the same [NvBlockNeeds](#).

11.5.4.2 NvBlockNeeds

The requested NVRAM Block configuration of the *NVRAM Manager* is described by the [NvBlockNeeds](#) of the [NvBlockDescriptor](#).

This information can be evaluated during ECU configuration similar to the [NvBlockNeeds](#) of an atomic software component or a BSW module. For further details see section [7.11.3](#).

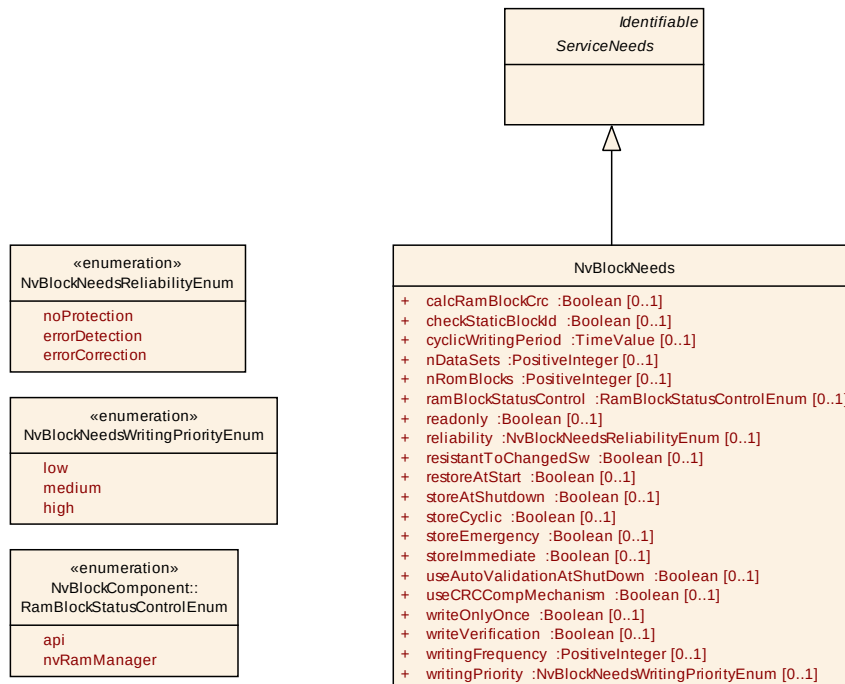


Figure 11.9: **NvBlockNeeds**

[constr_1308] Existence of [NvBlockNeeds.cyclicWritingPeriod](#) [The attribute [NvBlockNeeds.cyclicWritingPeriod](#) shall exist if and only if the attribute [NvBlockNeeds.storeCyclic](#) exists and its value is set to `true`.]()

Class	NvBlockNeeds			
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
Note	Specifies the abstract needs on the configuration of a single NVRAM Block.			
Base	ARObject, Identifiable , MultilanguageReferrable , Referrable , ServiceNeeds			
Attribute	Datatype	Mul.	Kind	Note
calcRamBlockCrc	Boolean	0..1	attr	Defines if CRC (re)calculation for the permanent RAM Block is required.
checkStaticBlockId	Boolean	0..1	attr	Defines if the Static Block Id check shall be enabled.
cyclicWritingPeriod	TimeValue	0..1	attr	This represents the period for cyclic writing of NvData to store the associated RAM Block.
nDataSets	PositiveInteger	0..1	attr	Number of data sets to be provided by the NVRAM manager for this block. This is the total number of ROM Blocks and RAM Blocks.
nRomBlocks	PositiveInteger	0..1	attr	Number of ROM Blocks to be provided by the NVRAM manager for this block. Please note that these multiple ROM Blocks are given in a contiguous area.
ramBlockStatusControl	RamBlockStatusControlEnum	0..1	attr	This attribute defines how the management of the RAM Block status is controlled.
readonly	Boolean	0..1	attr	True: data of this NVRAM Block are write protected for normal operation (but protection can be disabled) false: no restriction
reliability	NvBlockNeedsReliabilityEnum	0..1	attr	Reliability against data loss on the non-volatile medium.
resistantToConfigurationChanges	Boolean	0..1	attr	Defines whether an NVRAM Block shall be treated resistant to configuration changes (true) or not (false). For details how to handle initialization in the latter case, please refer to the NVRAM specification.
restoreAtStartup	Boolean	0..1	attr	Defines whether the associated RAM Block shall be implicitly restored during startup by the basic software.
storeAtShutdown	Boolean	0..1	attr	Defines whether or not the associated RAM Block shall be implicitly stored during shutdown by the basic software.
storeCyclic	Boolean	0..1	attr	Defines whether or not the associated RAM Block shall be implicitly stored periodically by the basic software.
storeEmergency	Boolean	0..1	attr	Defines whether or not the associated RAM Block shall be implicitly stored in case of ECU failure (e.g. loss of power) by the basic software. If the attribute storeEmergency is set to true the associated RAM Block shall be configured to have immediate priority.
storeImmediate	Boolean	0..1	attr	Defines whether or not the associated RAM Block shall be implicitly stored immediately during or after execution of the according SW-C RunnableEntity by the basic software.

Attribute	Datatype	Mul.	Kind	Note
useAutoValidationAtShutdown	Boolean	0..1	attr	If set to true the RAM Block shall be auto validated during shutdown phase.
useCRCComparisonMechanism	Boolean	0..1	attr	If set to true the CRC of the RAM Block shall be compared during a write job with the CRC which was calculated during the last successful read or write job in order to skip unnecessary NVRAM writings.
writeOnlyOnce	Boolean	0..1	attr	Defines write protection after first write: true: This block is prevented from being changed/erased or being replaced with the default ROM data after first initialization by the software-component. false: No such restriction.
writeVerification	Boolean	0..1	attr	Defines if Write Verification shall be enabled for this NVRAM Block.
writingFrequency	PositiveInteger	0..1	attr	Provides the amount of updates to this block from the application point of view. It has to be provided in "number of write access per year".
writingPriority	NvBlockNeedsWritingPriorityEnum	0..1	attr	Requires the priority of writing this block in case of concurrent requests to write other blocks.

Table 11.9: NvBlockNeeds

Enumeration	NvBlockNeedsReliabilityEnum
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds
Note	Reliability against data loss on the non-volatile medium. These requirements give only a relative indication, for example on the required degree of redundancy for storage. They do, however, not specify by which means (e.g. software or hardware) the reliability is actually achieved.
Literal	Description
errorCorrection	Errors shall be corrected
errorDetection	Errors shall be detected
noProtection	Data need not to be handled with protection

Table 11.10: NvBlockNeedsReliabilityEnum

[constr_1310] Existence of attributes of meta-class [NvBlockNeeds](#) [If in the context of an [ApplicationSwComponentType](#) the attribute [SwcServiceDependency.serviceNeeds](#) is implemented by an [NvBlockNeeds](#) then the following attributes

- [NvBlockNeeds.storeCyclic](#)
- [NvBlockNeeds.cyclicWritingPeriod](#)

- `NvBlockNeeds.storeEmergency`
- `NvBlockNeeds.storeImmediate`

shall only exist if in the context of the same `SwcServiceDependency` a `SwcServiceDependency.assignedPort` exists that has the attribute `role` set to the value `NvDataPort`. `]()`

11.5.4.3 RAM Block and ROM Block

[TPS_SWCT_01145] `ramBlock` and the `romBlock` are described by a `VariableDataPrototype` and a `ParameterDataPrototype` `[` The `ramBlock` and the `romBlock` are described by a `VariableDataPrototype` and a `ParameterDataPrototype` which are typed by an `AutosarDataType`. `]()`

[TPS_SWCT_01146] `romBlock` is optional `[` The `romBlock` is optional. If a `romBlock` is configured the RTE copies the `romBlock` constants into the RAM Block in case of a block initialization notification (`NvMNotifyInitBlock`). `]()`

[TPS_SWCT_01147] No `romBlock` is configured `[` If there is no `romBlock` configured the connected software components are either required to offer this functionality by a proper implementation of block initialization notification or the NVRAM Block has to be configured, that no ROM Block is needed. `]()`

As a mitigation against a failed read operation from NV memory it is recommended to always define a `romBlock` with suitable initial values to ensure the proper initialization of the corresponding `ramBlock`.

In particular, for software-components that don't define a `PortPrototype` typed by the `ClientServerInterface` with the standardized `shortName` `NotifyInitBlock` [31] it may happen that the `ramBlock` might not be properly initialized in case of failure.

[constr_2012] Compatibility of `ImplementationDataTypes` used for `ramBlock` and `romBlock` `[`

The `ramBlock` and the `romBlock` shall have compatible `ImplementationDataTypes` to ensure, that the NVRAM Block default values in the ROM Block can be copied into the RAM Block.

`]()`

Additionally it is possible that RAM Block and ROM Block are defined to be able to calibrate or measurable. Preceding `SwDataDefProps` might be defined with the means of an `InstantiationDataDefProps`.

11.5.4.4 NvBlockDataMapping

[TPS_SWCT_01148] NvBlockDataMapping [The meta-class `NvBlockDataMapping` specifies the mapping of `VariableDataPrototypes` of the `NvBlockSwComponentType`'s ports (`PPortPrototypes` / `RPortPrototypes`) to `VariableDataPrototypes` inside the `RAM Block`.]()

This ensures a flexible but deterministic `NVRAM Block` memory structure given by the `ImplementationDataType` of the `ramBlock` and `romBlock` and its association to the `PortPrototypes` of the `NvBlockSwComponentType`.

[constr_2013] Compatibility of ImplementationDataTypes for NvBlockDataMapping [The `NvBlockDataMapping` is only valid if the `ImplementationDataType` of the referenced `VariableDataPrototype` or `ImplementationDataTypeElement` in the role `nvRamBlockElement` is compatible to the `ImplementationDataType` used to type the `VariableDataPrototype` aggregated by `NvBlockDataMapping` in the role `writtenNvData`, `writtenReadNvData`, or `readNvData`.]()

[constr_1285] Applicability of roles vs. PortPrototypes [The aggregation of `AutosarVariableRef` aggregated by `NvBlockDataMapping` in the roles `writtenNvData`, `writtenReadNvData`, or `readNvData` is subject to limitation depending on the applicable subclass of `PortPrototype`:

- The role `writtenNvData` shall only be used if the corresponding `PortPrototype` is a `RPortPrototype`
- The role `writtenReadNvData` shall only be used if the corresponding `PortPrototype` is a `PRPortPrototype`
- The role `readNvData` shall only be used if the corresponding `PortPrototype` is a `PPortPrototype`

]()

But nevertheless it is valid, that not all `ImplementationDataTypeElements` within the `VariableDataPrototype` aggregated by `NvBlockDescriptor` in the role `ramBlock` are mapped to a `VariableDataPrototype` located in a `PortPrototype`.

This enables to have fill elements or logistic data in the `NVRAM Block` which are not accessed by software components. This is exemplified by the element `x` in Figure 11.10.

Please note that the `VariableDataPrototype` located in the `PortPrototype`, in the vast majority of cases, will be typed by an `ApplicationDataType` which in turn (at least before the actual code generation starts) finally shall have a mapping to an `ImplementationDataType`. This aspect is explained in chapter 5.2.2.

[TPS_SWCT_01659] Mapping of VariableDataPrototype to a NvBlockDescriptor [There are three ways to map a `VariableDataPrototype` (i.e. `Nv-`

`NvDataInterface.nvData` in the context of a specific `PortPrototype`) to either an `NvBlockDescriptor.ramBlock` or a *sub-element* thereof:

- `NvDataInterface.nvData` is directly and completely mapped, i.e. `AutosarVariableRef.autosarVariable` shall exist and `autosarVariable.targetDataPrototype` shall refer to the `NvDataInterface.nvData`.
- **Every** *leaf* element of `NvDataInterface.nvData` is mapped individually. This means that either
 - `AutosarVariableRef.autosarVariableInImplDatatype` shall exist and `autosarVariableInImplDatatype.targetDataPrototype` shall refer to the respective *leaf* element of `NvDataInterface.nvData`.
 - `AutosarVariableRef.autosarVariable` shall exist and `autosarVariable.targetDataPrototype` shall refer to the respective *leaf* element of `NvDataInterface.nvData`.

In other words: the mapping shall be defined either via the used `ImplementationDataType` or else via the used `ApplicationDataType`.

- A *sub-element* of `NvDataInterface.nvData` - which is **not** a *leaf* element - may be directly mapped and consequently **all** the *leaf* elements of the respective *sub-element* of `NvDataInterface.nvData` are **indirectly mapped** as well. This means that
 - `AutosarVariableRef.autosarVariableInImplDatatype` shall exist and `autosarVariableInImplDatatype.targetDataPrototype` shall refer to the *sub-element* element of `NvDataInterface.nvData`.
 - `AutosarVariableRef.autosarVariable` shall exist and `autosarVariable.targetDataPrototype` shall refer to the *sub-element* element of `NvDataInterface.nvData`.

In other words: the mapping shall be defined either via the used `ImplementationDataType` or else via the used `ApplicationDataType`.

]()

Please note that a mixing of **mutually exclusive** mappings for entire *sub-elements* or *leaf* elements as described by [TPS_SWCT_01659] is positively supported (see Figure 11.10).

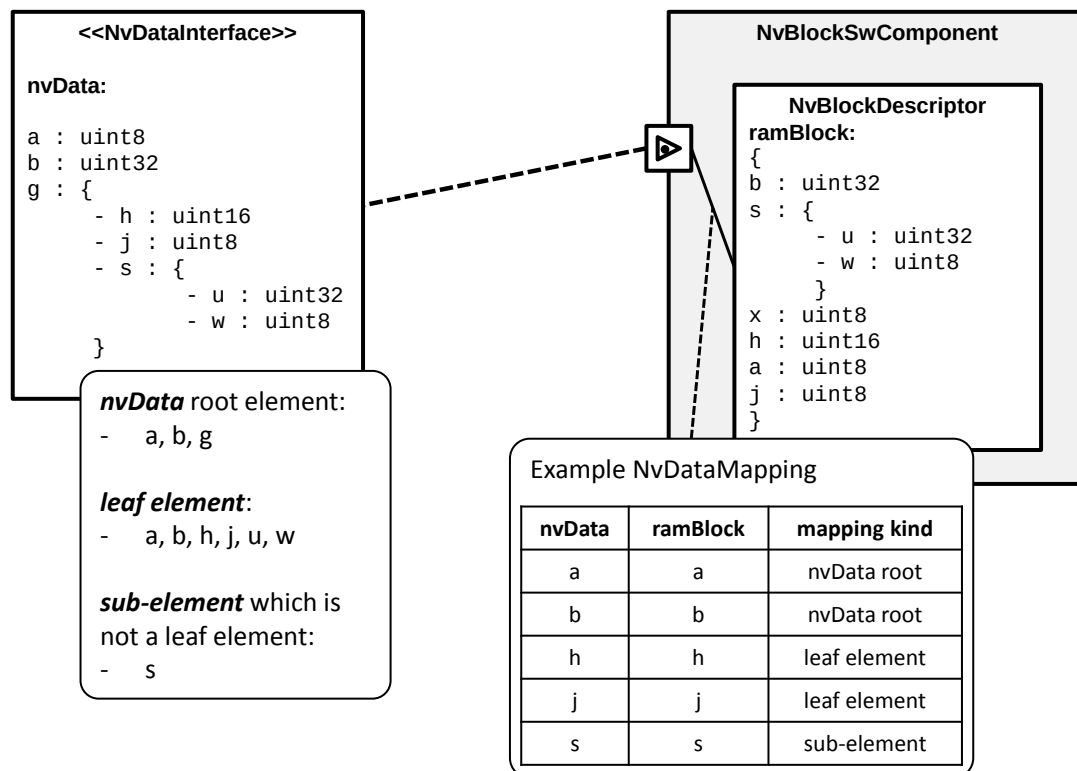


Figure 11.10: Example `NvBlockDataMapping` to explain [TPS_SWCT_01659]

[constr_1395] `NvBlockDataMapping` shall be complete [If an `NvBlockDataMapping` refers to *sub-elements* or *leaf* elements of the `NvDataInterface.nvData` in the context of a particular `PortPrototype` then **all remaining sub-elements** or *leaf* elements **shall effectively be mapped** according to [TPS_SWCT_01659] by means of a collection of `NvBlockDataMappings`.]()

[constr_1403] `NvBlockDataMappings` to a given `nvData` shall be unambiguous [If an `NvBlockDataMapping` exists that **directly** and **completely** maps a specific `NvDataInterface.nvData` in the context of a particular `PortPrototype` then **no** other `NvBlockDataMapping` which maps sub-elements of the `NvDataInterface.nvData` shall exist.]()

The interaction with AUTOSAR services is centrally defined in the context of the `SwcServiceDependency`. The latter gathers a collection of `PortPrototypes` by means of `RoleBasedPortAssignments` that implement a closely related service functionality.

In the specific case of interaction between `AtomicSwComponentType` and `NvBlockSwComponentType` (as described by [TPS_SWCT_02503]), there are `PortPrototypes` referenced by a `RoleBasedPortAssignment` with attribute `RoleBasedPortAssignment.role` set to `NvDataPort`. These `PortPrototypes` contain the collected *Nv Data* of the service use case.

Furthermore, there is the possibility to receive notifications when the writing of the mapped NV Block to the NvRam is finished.

In order to be able to properly assign such a notification to the content of the related *Nv Data PortPrototypes* in the scope of the same *SwcServiceDependency* it is necessary that the *Nv Data* of **all** these *PortPrototypes* is mapped to the **same** *Nv Block* (because the notifications are created **per block**).

This motivates the existence of [constr_1404]:

[constr_1404] All *NvDataInterface.nvData* of *PortPrototypes* in the context of a specific *SwcServiceDependency* shall be mapped to the same *NvBlock-Descriptor* [In the context of a given *SwcServiceDependency* (which, in turn, is owned by an *AtomicSwComponentType*), **all** *NvDataInterface.nvData* of *Port-Prototypes* referenced by a *RoleBasedPortAssignment* with attribute *RoleBasedPortAssignment.role* set to *NvDataPort* shall be connected (either directly or via the definition of suitable *PortInterfaceMappings*) to *NvDataInterface.nvData* (on the side of the *NvBlockSwComponentType*) that are **completely mapped** (via *NvBlockDataMappings*) to the identical *NvBlockDescriptor.ramBlock*.]()

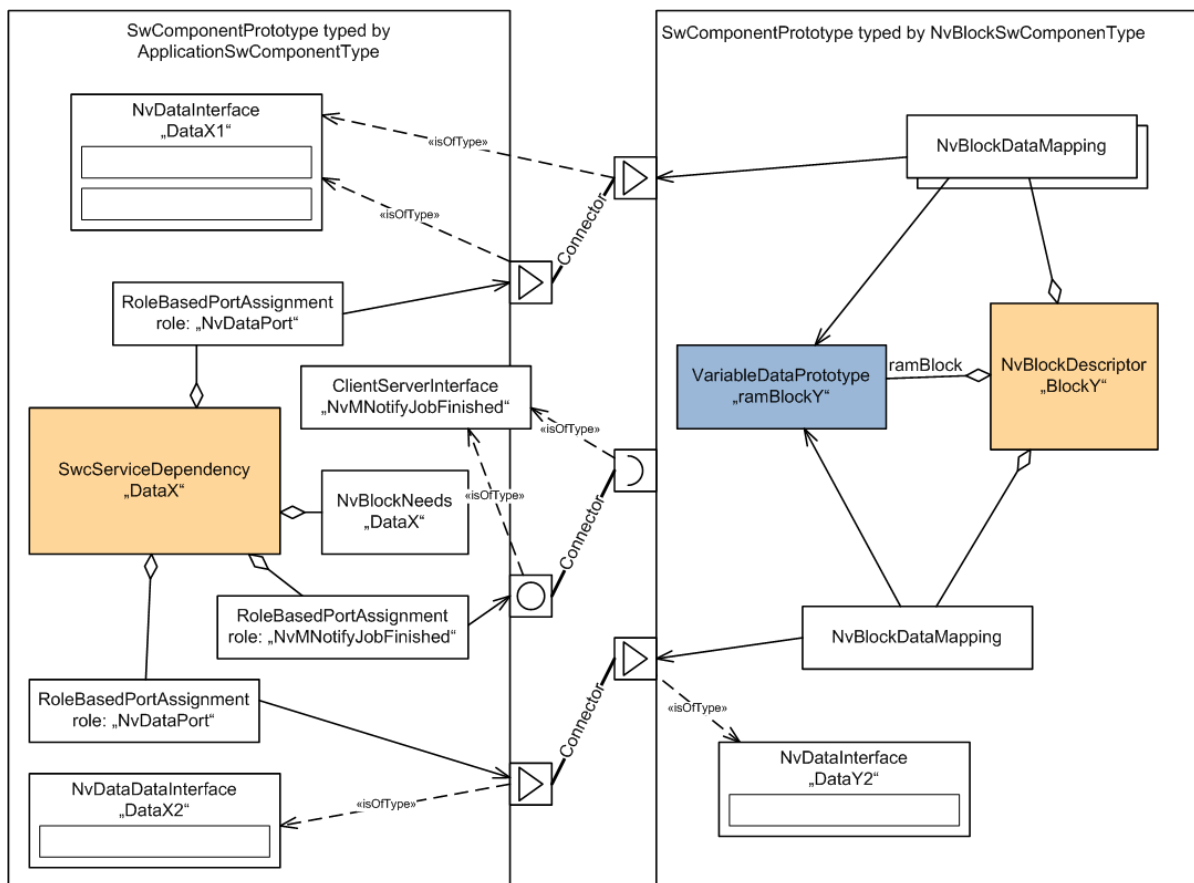


Figure 11.11: Visualization of the statement made by [constr_1404]

The statement made by [constr_1404] is visualized in Figure 11.11. The context-defining model elements, i.e. *SwcServiceDependency* owned by the *Atomic-*

`SwComponentType` as well as `NvBlockDescriptor` owned by the `NvBlockSwComponentType`, are colored in light orange.

The diagram is focused on the `NvBlockDescriptor.ramBlock`. As stressed by [constr_1404], all *Nv Data* provided by the `PortPrototypes` referenced by the specific `SwServiceDependency` finally ends up in the one depicted `ramBlock` (colored in blue).

Please note that the graphical representation of the `NvBlockDataMapping` in Figure 11.11 has been simplified for the sake of clarity.

Class	NvBlockDataMapping			
Package	M2::AUTOSARTemplates::SWComponentTemplate::NvBlockComponent			
Note	Defines the mapping between the <code>VariableDataPrototypes</code> in the <code>NvBlockComponents</code> ports and the <code>VariableDataPrototypes</code> of the RAM Block. The data types of the referenced <code>VariableDataPrototypes</code> in the ports and the referenced sub-element (inside a <code>CompositeDataType</code>) of the <code>VariableDataPrototype</code> representing the RAM Block shall be compatible.			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
nvRamBlockElement	AutosarVariableRef	1	aggr	Reference to a <code>VariableDataPrototype</code> of a RAM Block.
readNvData	AutosarVariableRef	0..1	aggr	Reference to a <code>VariableDataPrototype</code> of a <code>pPort</code> of the <code>NvBlockComponent</code> providing read access to the RAM Block. If there is no <code>PortPrototype</code> providing read access (write-only) the reference can be omitted.
writtenNvData	AutosarVariableRef	0..1	aggr	Reference to a <code>VariableDataPrototype</code> of a <code>rPort</code> of the <code>NvBlockComponent</code> providing write access to the RAM Block. If there is no port providing write access (read-only) the reference can be omitted.
writtenReadNvData	AutosarVariableRef	0..1	aggr	Reference to a <code>VariableDataPrototype</code> of a <code>PRPortPrototype</code> of the <code>NvBlockSwComponentType</code> providing write and read access to the RAM Block.

Table 11.11: NvBlockDataMapping

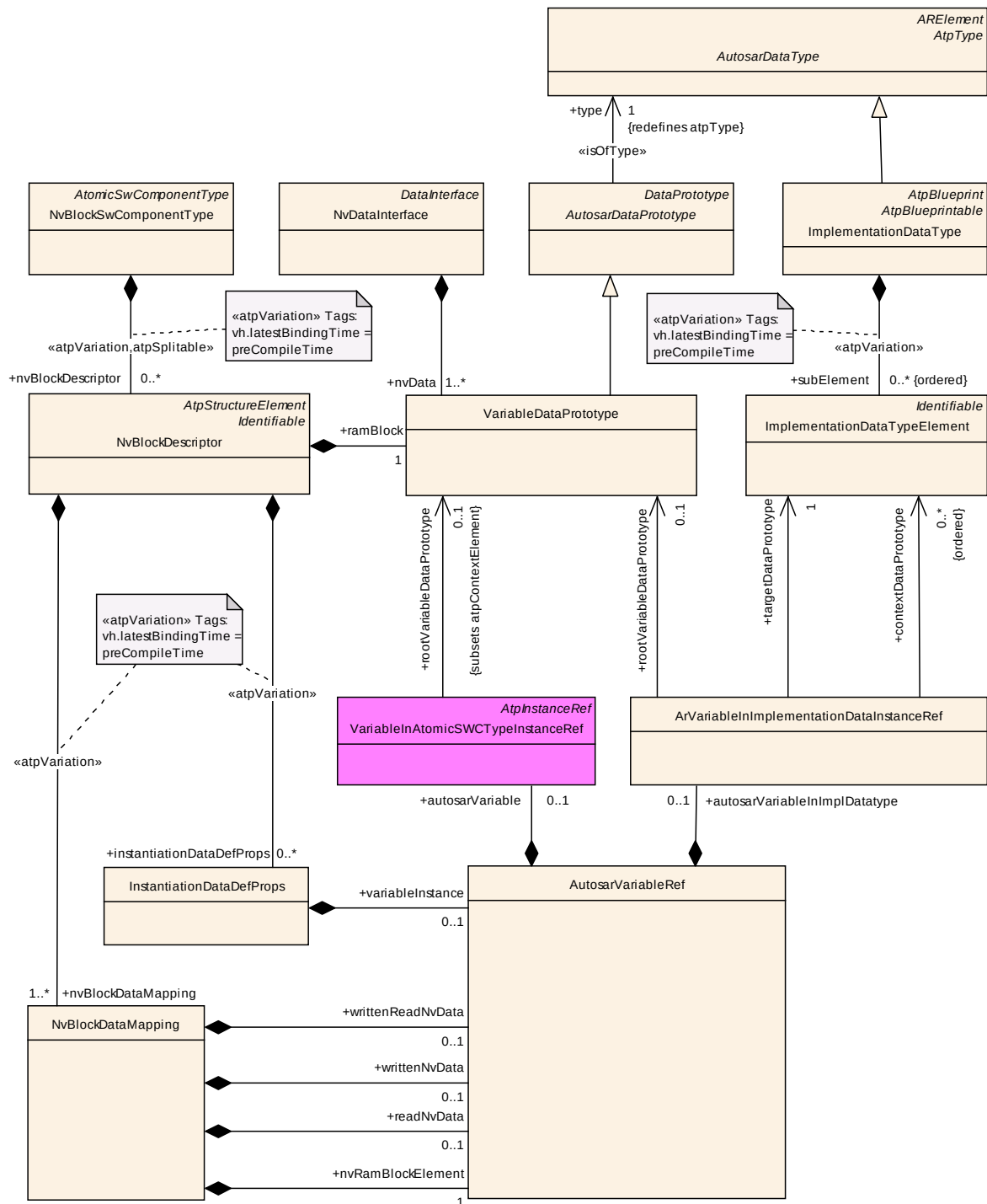


Figure 11.12: NvBlockToPortMapping and InstantiationDataDefProps

11.5.4.5 Client Server Ports

[TPS_SWCT_01149] RoleBasedPortAssignment of NvBlockDescriptor [The clientServerPort of the NvBlockDescriptor describes which client/server PortPrototype of the NvBlockSwComponentType serves for which purpose. The

`role` specifies if the port serves for block-related services, administrative services or notification. `⌋()`

[constr_2014] Limitation of RoleBasedPortAssignment.role in NvBlockDescriptors | The `role` has to be set to a valid name of the *Standardized AUTOSAR Interface* used for the *NVRAM Manager* e.g. *NvMNotifyJobFinished* or *NvMNotifyInitBlock*. |()

In case of notifications one common callback function is provided by the RTE for each individual kind of notification defined by the `role`.

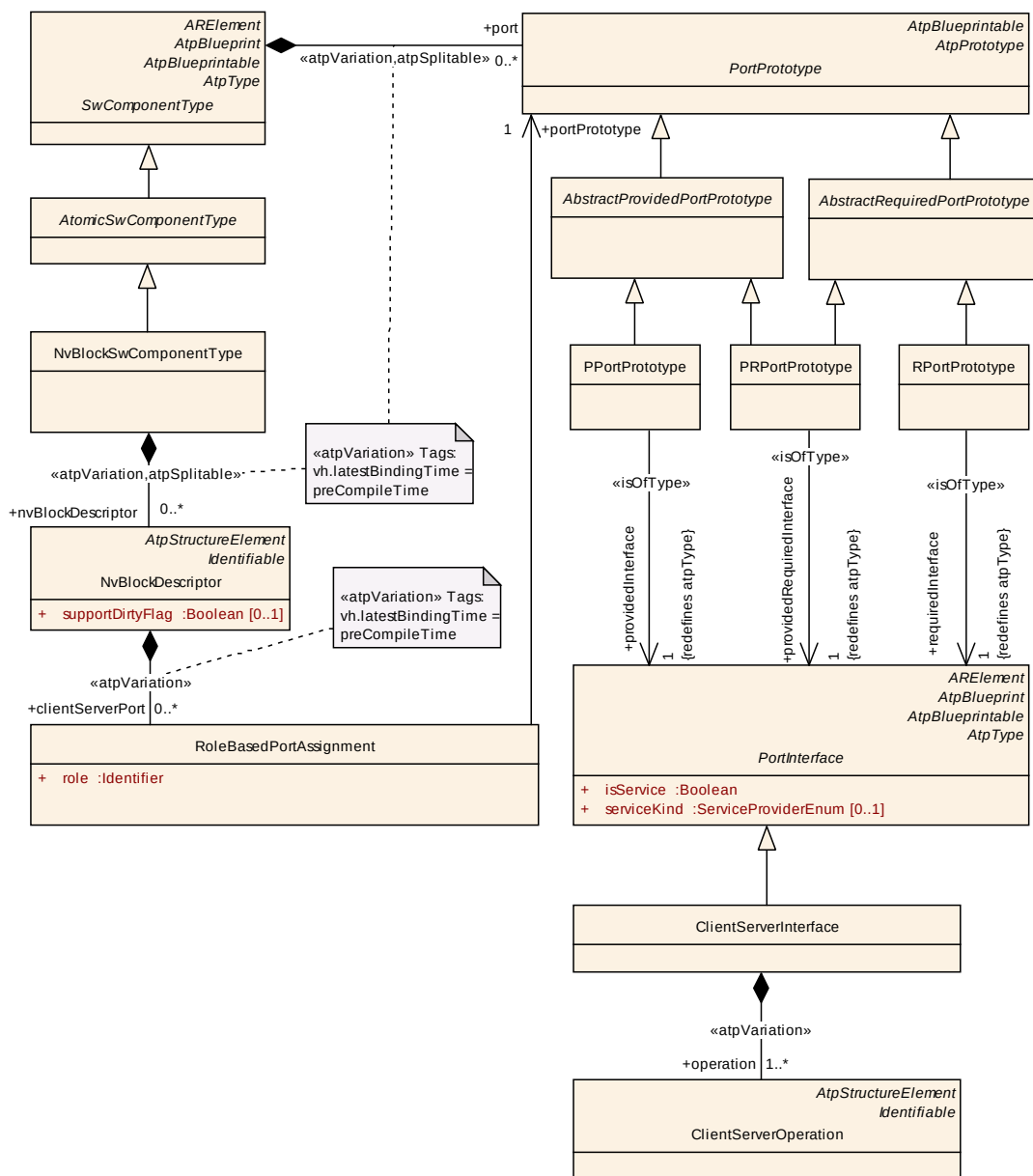


Figure 11.13: NvBlockNotification

11.5.5 SwcInternalBehavior of an NvBlockSwComponentType

[TPS_SWCT_01150] InternalBehavior of a NvBlockSwComponentType to enable access to the NVRAM Block management API [In general, the `InternalBehavior` of a `NvBlockSwComponentType` is only used for a limited scope.

The main use case is that the `NvBlockSwComponentType` defines `PPortPrototypes` typed by a `ClientServerInterface` to enable access to the NVRAM Block management API.

To enable the configuration of the server invocation in the RTE's ECU configuration, the `NvBlockSwComponentType` needs to provide the following model elements:

- `OperationInvokedEvents`
- server `RunnableEntity`
- `PortDefinedArgumentValues` to define the NVRAM Block ID which has to be passed to the NvM

In addition to the above list further model elements may qualify; the details are explained in [\[TPS_SWCT_01584\]](#). `]()`

[TPS_SWCT_01584] InternalBehavior of a NvBlockSwComponentType for implementing a writing strategy [For the use case that `NvBlockDescriptors` exists that aggregate `NvBlockNeeds` which, in turn, define particular NV data writing strategies (by defining any of the attributes `storeAtShutdown`, `storeImmediate`, `storeEmergency`, or `storeCyclic`) the `InternalBehavior` of a `NvBlockSwComponentType` needs to support further model elements.

Particularly, In addition to the model elements listed in [\[TPS_SWCT_01150\]](#), the following list of model elements can be used in the `InternalBehavior` of a `NvBlockSwComponentType` for implementing writing strategies:

- `TimingEvents` (which may include references to `ModeDeclarations` in the role `disabledMode`)
- `DataReceivedEvents` (which may include references to `ModeDeclarations` in the role `disabledMode`)
- `SwcModeSwitchEvents`
- `RunnableEntities`

`](RS\_SWCT\_03225)`

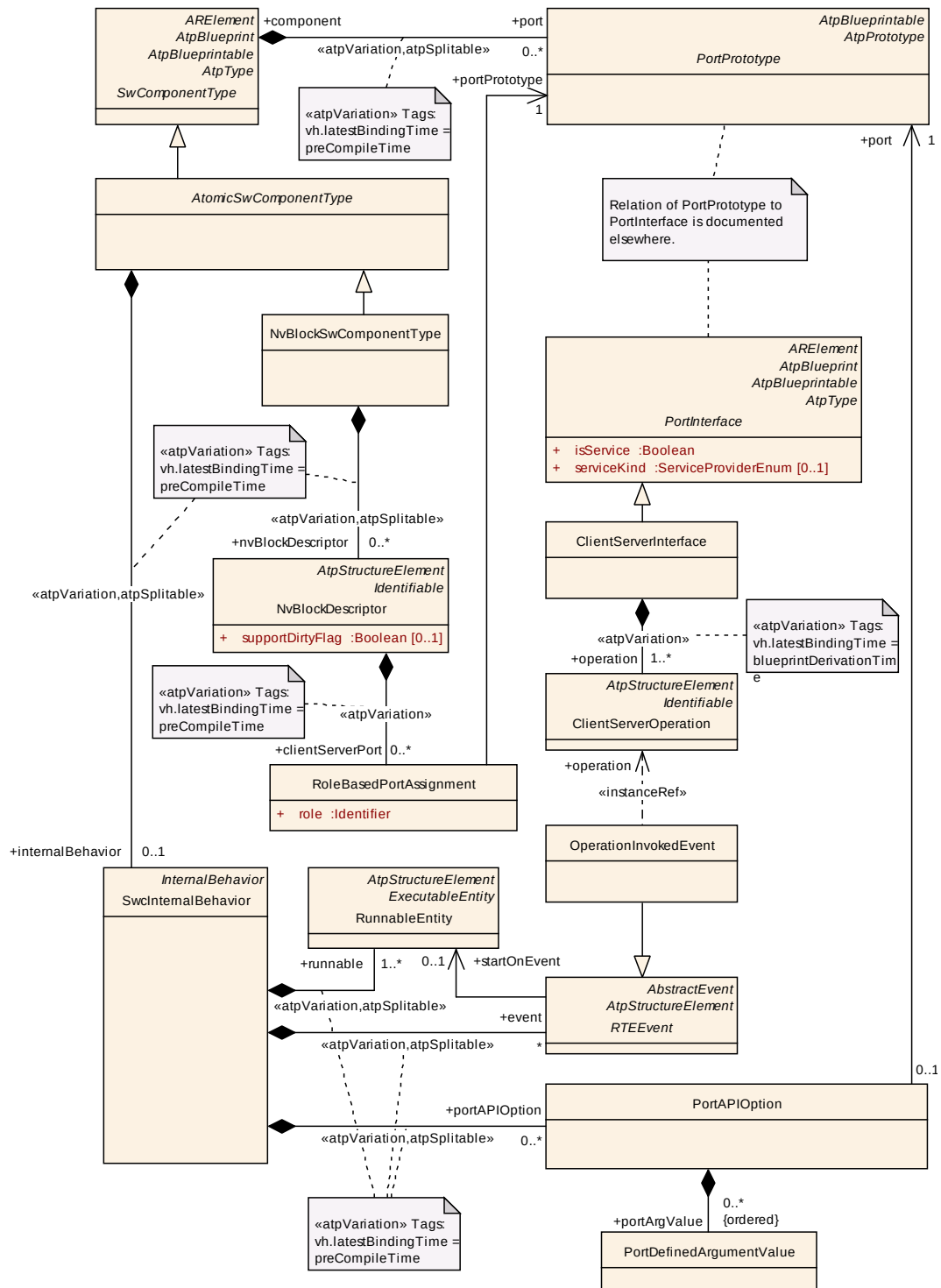


Figure 11.14: **NvBlockSwComponentType** and **SwcInternalBehavior**

[TPS_SWCT_01152] **InternalBehavior** does not have further attributes [It is not expected, that such **InternalBehavior** do have further attributes like **ExclusiveAreas**, per-instance memory or inter-runnable variables, etc.]()

[TPS_SWCT_01151] **RunnableEntity**s do not have further attributes [The same condition exists for the **RunnableEntity**s of such **InternalBehavior** which shall

not define further attributes, e.g. data access points (implemented by means of references from `SwcInternalBehavior` to `VariableAccess`) or `ServerCallPoints`.
]()

[constr_1234] Value of `RunnableEntity.symbol` [The value of a `RunnableEntity.symbol` owned by an `NvBlockSwComponentType` that is triggered by an `OperationInvokedEvent` shall only be taken from the set of API names associated with the `NvM`.]()

For example, `RunnableEntity.symbol` owned by an `NvBlockSwComponentType` could rightfully be set to `NvM_ReadBlock` [31] but an arbitrary value like `ReadThisBlock` is not permitted.

The rationale for [constr_1234] is that the `RunnableEntity`s that are triggered by an `OperationInvokedEvent` are not existing as such but are mapped to the respective function calls of the `NvM`. For more details of how this mapping can be achieved please refer to [7].

Please note that no restriction applies for the value of attribute `RunnableEntity.symbol` of any `RunnableEntity` owned by an `NvBlockSwComponentType` that is triggered by an `RTEEvent` other than `OperationInvokedEvent`.

[constr_2015] Limitation of `SwcInternalBehavior` of a `NvBlockSwComponentType` [The `SwcInternalBehavior` of a `NvBlockSwComponentType` is only permitted to define

- `OperationInvokedEvents`
- `RunnableEntity`s triggered by `OperationInvokedEvents` (server `RunnableEntity`s)
- `RunnableEntity`s which defines only the mandatory attributes `symbol` and `canBeInvokedConcurrently`
- `PortAPIOptions` defining `PortDefinedArgumentValues`
- `TimingEvents` (which may include references to `ModeDeclarations` in the role `disabledMode`)
- `DataReceivedEvents` (which may include references to `ModeDeclarations` in the role `disabledMode`)
- `SwcModeSwitchEvents`
- `RunnableEntity`s triggered by `TimingEvents`
- `RunnableEntity`s triggered by `DataReceivedEvents`
- `RunnableEntity`s triggered by `SwcModeSwitchEvents`

]()

[constr_1309] Existence of `NvBlockDescriptor.timingEvent` [The attribute `NvBlockDescriptor.timingEvent` shall exist if and only if the `NvBlockDescriptor.nvBlockNeeds.storeCyclic` exists and is set to the value `true`.]()

Note that there is a conceptual connection between the values of the two attributes `NvBlockDescriptor.timingEvent.period` and `SwcServiceDependency.serviceNeeds.cyclicWritingPeriod`.

Specifically, the `SwcServiceDependency.serviceNeeds.cyclicWritingPeriod` represents a requirement and the `NvBlockDescriptor.timingEvent.period` is supposed to fulfill the requirement.

[TPS_SWCT_01585] Relevance of `NvBlockDescriptor.timingEvent.period` [For any given `NvBlockDescriptor`, the value of the attribute `NvBlockDescriptor.nvBlockNeeds.cyclicWritingPeriod` shall be ignored and the value of `NvBlockDescriptor.timingEvent.period` shall be taken to specify the effective writing frequency for cyclic storage.]([RS_SWCT_03225](#))

12 Software Component Documentation

AUTOSAR supports documentation of software component types by adopting the principles of ASAM-FSX [43] Standard to AUTOSAR. With AUTOSAR Release 4.0 the AUTOSAR XML schema 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 [12].

[TPS_SWCT_01062] Documentation of software-components [As shown in figure 12.1, the documentation of a software component is composed of several chapters. Some chapters are predefined, describing the component from the perspective of different activities performed on the component like testing it (`swTestDesc`), maintaining it (`swMaintenanceNotes`), calibrating it (`swCalibrationNotes`) or performing diagnostic (`swDiagnosticsNotes`) on the component.]([RS_SWCT_02110](#), [RS_SWCT_03230](#))

Two other predefined chapters describe the component (`swFeatureDesc`) and define its physical functionality (`swFeatureDef`). In order to describe additional aspects of a software component, an arbitrary number of free chapters can be defined.

The predefined chapters typically provide informal guideline (e.g., recommendation) or documentation. Formal information can be captured using special data groups [12] or annotating documentation construct with semantic information. This could be used to extend the predefined chapters or in separate free chapters.

Note that the documentation of a software component can be stored in a different file than the component itself (i.e., it is `<<atpSplitable>>` from the component).

Each of the predefined and free chapters follows the `<<atpVariation>>` stereotype to support variant handling (see [12]) on the documentation at the chapter level. These variation points have a post-build as latest binding time, because the decision to include or exclude a chapter as well as the decision which variant of this chapter should be included can be made when the component has been built.

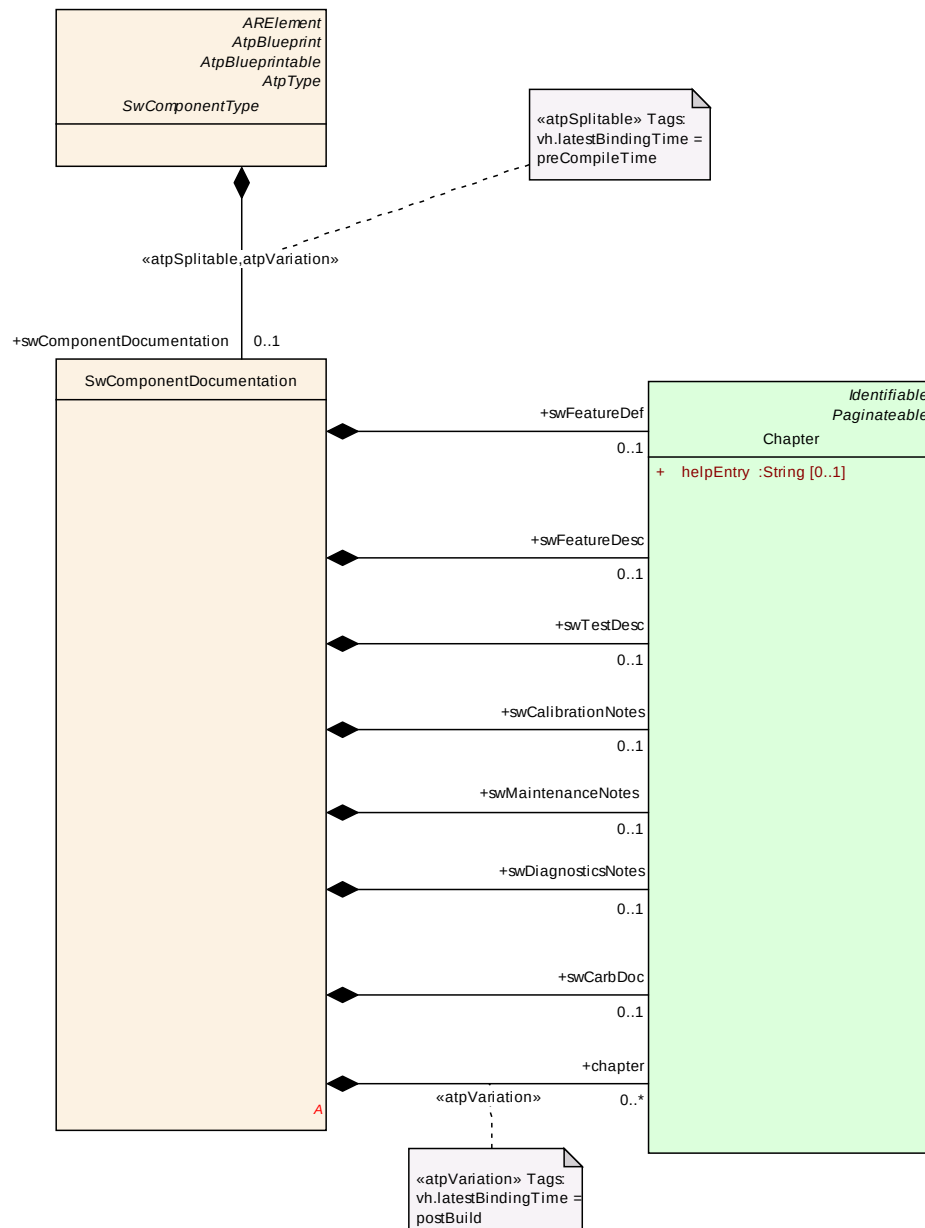


Figure 12.1: Software component documentation

Class	SwComponentDocumentation			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SoftwareComponent Documentation			
Note	This class specifies the ability to write dedicated documentation to a component type according to ASAM FSX.			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note

Attribute	Datatype	Mul.	Kind	Note
chapter	Chapter	*	aggr	These chapters provide additional information about the software component that do not fit in the other chapters. Note that this is subject to variation because Chapter aggregations in the role chapter are variant within the documentation in general. Stereotypes: atpVariation Tags: vh.latestBindingTime=postBuild xml.roleElement=true; xml.roleWrapperElement=false; xml.sequenceOffset=100; xml.typeElement=false
swCalibrationNotes	Chapter	0..1	aggr	This element contains calibration instructions and hints for a calibration engineer. Tags: xml.roleElement=true; xml.sequenceOffset=60; xml.typeElement=false
swCarbDoc	Chapter	0..1	aggr	This element records the documentation requested by CARB. Tags: xml.roleElement=true; xml.sequenceOffset=80; xml.typeElement=false
swDiagnosticsNotes	Chapter	0..1	aggr	This element contains general information about diagnostics issues within the component. Tags: xml.roleElement=true; xml.sequenceOffset=75; xml.typeElement=false
swFeatureDef	Chapter	0..1	aggr	This element contains the definition of the physical functionality of this software component. This definition is more or less formal and is intended to be delivered from modeling tools. Tags: xml.roleElement=true; xml.sequenceOffset=20; xml.typeElement=false
swFeatureDesc	Chapter	0..1	aggr	This element contains the textual description of the software functionality of this software component. Expert should write this description. Tags: xml.roleElement=true; xml.sequenceOffset=30; xml.typeElement=false
swMaintenanceNotes	Chapter	0..1	aggr	This element contains information regarding the software maintenance of the component. Tags: xml.roleElement=true; xml.sequenceOffset=70; xml.typeElement=false
swTestDesc	Chapter	0..1	aggr	This element contains suggestions and hints for the test of the software functionality of this software component. Tags: xml.roleElement=true; xml.sequenceOffset=50; xml.typeElement=false

<i>Attribute</i>	<i>Datatype</i>	<i>Mul.</i>	<i>Kind</i>	<i>Note</i>

Table 12.1: SwComponentDocumentation

13 Rapid Prototyping Scenarios

13.1 Definition of Rapid Prototyping Scenario

A Rapid Prototyping Scenario consist out of two main aspects: The description of the `byPassPoints` and the relation to a `rptHook`. A Rapid Prototyping Scenario is structured by means of `RptContainers`. The correct usage of `RptContainer` structure is described in 13.2.

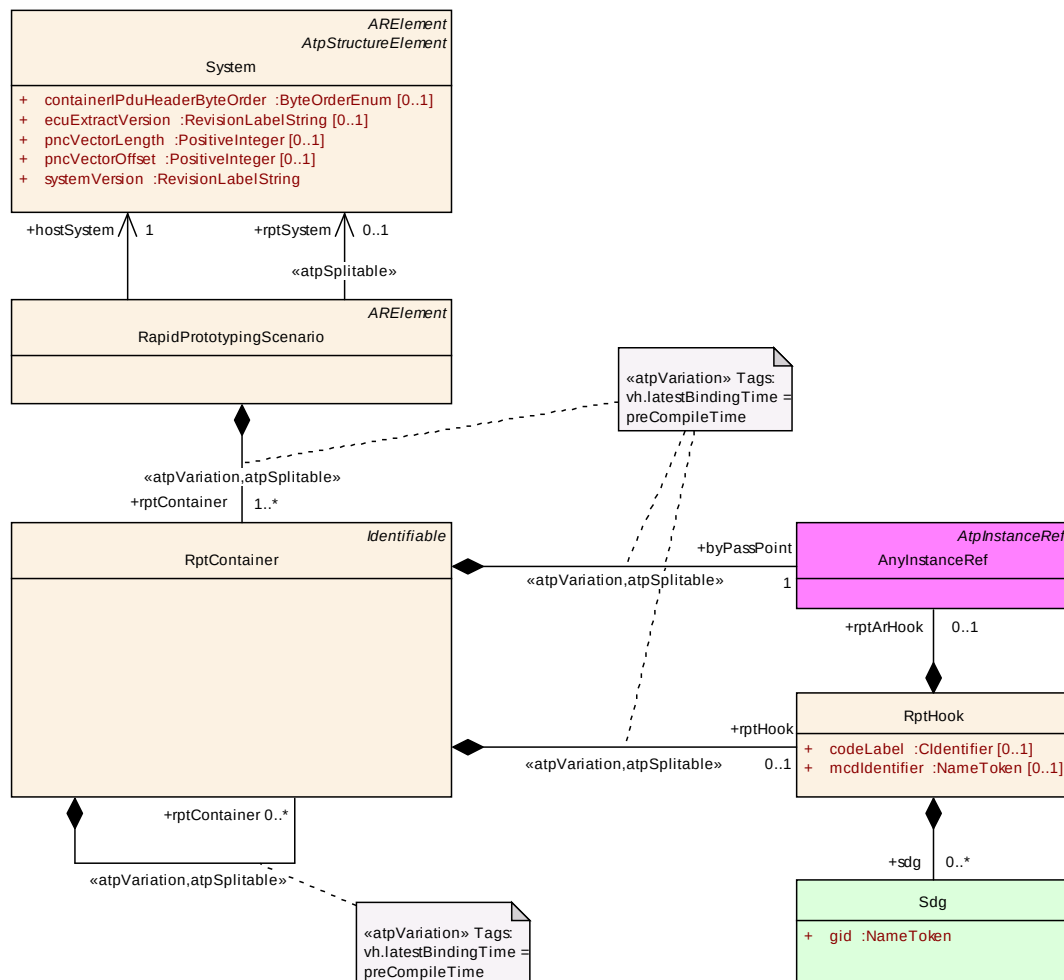


Figure 13.1: Rapid Prototyping Scenario

Class	RapidPrototypingScenario			
Package	M2::AUTOSARTemplates::SWComponentTemplate::RPTScenario			
Note	This meta class provides the ability to describe a Rapid Prototyping Scenario. Such a Rapid Prototyping Scenario consist out of two main aspects, the description of the byPassPoints and the relation to an rptHook. Tags: atp.recommendedPackage=RapidPrototypingScenarios			
Base	ARElement , ARObject , CollectableElement , Identifiable , MultilanguageReferrable , PackageableElement , Referrable			
Attribute	Datatype	Mul.	Kind	Note
hostSystem	System	1	ref	System which describes the software components of the host ECU.
rptContainer	RptContainer	1..*	aggr	Top-level rptContainer definitions of this specific rapid prototyping scenario. Stereotypes: atpSplitable; atpVariation Tags: atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime
rptSystem	System	0..1	ref	System which describes the rapid prototyping algorithm in the format of AUTOSAR Software Components. Stereotypes: atpSplitable Tags: atp.Splitkey=rptSystem

Table 13.1: RapidPrototypingScenario

Class	RptContainer			
Package	M2::AUTOSARTemplates::SWComponentTemplate::RPTScenario			
Note	This meta class defines a byPassPoint and the relation to a rptHook. Additionally it may contain further rptContainers if the byPassPoint is not atomic. For example a byPassPoint refereing to a RunnableEntity may contain rptContainers referring to the data access points of the RunnableEntity. The RptContainer structure on M1 shall follow the M1 structure of the Software Component Descriptions. The category attribute denotes which level of the Software Component Description is annotated.			
Base	ARObject , Identifiable , MultilanguageReferrable , Referrable			
Attribute	Datatype	Mul.	Kind	Note
byPassPoint	AtpFeature	1	iref	byPassPoint describes the required preparation of the host ECU. At a byPassPoint the host ECU shall be capable to communicate with a RPT System in order to support the execution of the rapid prototyping algorithms with the original data calculated by the host system and to replace dedicated results of the host system by the results of the rapid prototyping algorithm. Stereotypes: atpSplitable; atpVariation Tags: atp.Splitkey=byPassPoint vh.latestBindingTime=preCompileTime

Attribute	Datatype	Mul.	Kind	Note
rptContainer	RptContainer	*	aggr	Sub-level rptContainer definitions of this specific rapid prototyping scenario. Stereotypes: atpSplitable; atpVariation Tags: atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime
rptHook	RptHook	0..1	aggr	The rptHook describes the link between a byPassPoint and the rapid prototyping algorithm. Stereotypes: atpSplitable; atpVariation Tags: atp.Splitkey=rptHook, variationPoint.shortLabel vh.latestBindingTime=preCompileTime

Table 13.2: RptContainer

Class	RptHook			
Package	M2::AUTOSARTemplates::SWComponentTemplate::RPTScenario			
Note	This meta class provide the ability to describe a rapid prototyping hook. This can either be described by an other AUTOSAR system with the category RPT_SYSTEM or as a non AUTOSAR software.			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
codeLabel	CIdentifier	0..1	ref	This attribute provides a code label which is used in the implementation of the hook. For example this can be an C function name or the name of data definition.
mcdIdentifier	NameToken	0..1	attr	This attribute provides an identifier which shall be used in a MCD System to display the Rpt Hook.
rptArHook	AtpFeature	0..1	iref	This describes the hook with the means of another AUTOSAR system.
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.

Table 13.3: RptHook

[TPS_SWCT_02046] [byPassPoint](#) specifies the rapid prototyping capability [The [byPassPoints](#) are used to describe the preparation of the host ECU. At the [byPassPoints](#) the host ECU shall be capable to communicate with a RPT System in order to support the execution of the rapid prototyping algorithms with the original data calculated by the host system and to replace dedicated results of the host system by the results of the rapid prototyping algorithm.] ([RS_SWCT_03280](#))

[TPS_SWCT_02047] [rptHook](#) specifies the link to rapid prototyping algorithm [The [rptHook](#) describes the link between the [byPassPoint](#) and the rapid prototyping algorithm. If the rapid prototyping algorithm is described as an AUTOSAR Soft-

ware Component the `rptArHook` reference is applicable. Otherwise the definition of a `codeLabel` and optionally `mcdIdentifier` shall be used. [\]\(RS_SWCT_03280\)](#)

In order to describe an RPT system as AUTOSAR software component a `System` with the `category` `RPT_SYSTEM` shall be defined.

[constr_2054] Valid targets of `rptSystem` [\[](#) The `System` referenced in the role `rptSystem` shall be of `category` `RPT_SYSTEM`. [\]\(](#)

13.2 Usage of RptContainers on M1

The `RptContainer` structure on M1 shall follow the M1 structure of the Software Component Descriptions. The `category` attribute denotes which level of the Software Component Description is annotated.

The following values of the attribute `category` are predefined by the AUTOSAR standard:

Category	Meaning	Specific properties
SW_COMPONENT_PROTOTYPE	Adds one <code>SwComponentPrototype</code> to an Rapid Prototyping Scenario.	The <code>byPassPoint</code> and <code>rptArHook</code> shall reference a <code>SwComponentPrototypes</code> .
DATA_PROTOTYPE	Adds one instance of a <code>DataPrototype</code> to an Rapid Prototyping Scenario.	The <code>byPassPoint</code> and <code>rptArHook</code> shall reference a <code>DataPrototype</code> instances in <code>PortPrototypes</code>
RUNNABLE_ENTITY	Adds one <code>RunnableEntity</code> to an Rapid Prototyping Scenario.	The <code>byPassPoint</code> and <code>rptArHook</code> shall reference a <code>RunnableEntity</code> instances.
ACCESS_POINTS	Adds one <code>VariableAccess</code> , <code>ParameterAccess</code> , <code>ServerCallPoint</code> , <code>AsynchronousServerCallResultPoint</code> , <code>InternalTriggeringPoint</code> , <code>ModeSwitchPoint</code> , <code>ModeAccessPoint</code> or <code>ExternalTriggeringPoint</code> to an Rapid Prototyping Scenario.	The <code>byPassPoint</code> and <code>rptArHook</code> shall reference a <code>VariableAccess</code> , <code>ParameterAccess</code> , <code>ServerCallPoint</code> , <code>AsynchronousServerCallResultPoint</code> , <code>InternalTriggeringPoint</code> , <code>ModeSwitchPoint</code> , <code>ModeAccessPoint</code> or <code>ExternalTriggeringPoint</code> instances.

Table 13.4: Category of RptContainers

[constr_2055] Valid targets of `byPassPoint` and `rptHook` reference [\[](#) Depending on the `category` value the targets of `byPassPoint` and `rptHook` references are restricted according table [13.4](#). [\]\(](#)

Hereby, the following semantic applies:

[TPS_SWCT_02048] Implicit `SwComponentPrototype` selection for Rapid Prototyping Scenario [\[](#) If a `SwComponentPrototype` is referenced in the role `byPassPoint` by a `RptContainer` without further “Sub” `rptContainer` all RTE Interfaces of the `AtomicSwComponentType` shall be able to support a connection to a `rptHook`. [\]\(RS_SWCT_03280\)](#)

[TPS_SWCT_02049] Implicit RunnableEntity selection for Rapid Prototyping Scenario [If a `RunnableEntity` is referenced in the role `byPassPoint` by a `RptContainer` without further “Sub” `rptContainer` all RTE Interfaces of the `RunnableEntity` shall be able to support a connection to a `rptHook`.]([RS_SWCT_03280](#))

[TPS_SWCT_02050] Explicit access point selection for Rapid Prototyping Scenario [If a `VariableAccess`, `ParameterAccess`, `ServerCallPoint`, `AsynchronousServerCallResultPoint`, `InternalTriggeringPoint`, `ModeSwitchPoint`, `ModeAccessPoint` or `ExternalTriggeringPoint` is referenced in the role `byPassPoint` by a `RptContainer` only RTE Interfaces related to the specific access point are required be able to support a connection to a `rptHook`.]([RS_SWCT_03280](#))

[TPS_SWCT_02051] Explicit DataPrototype selection for Rapid Prototyping Scenario [If a `DataPrototype` instances in a `PortPrototypes` is referenced in the role `byPassPoint` by a `RptContainer` only RTE Interfaces related to the specific `DataPrototype` are required be able to support a connection to a `rptHook`.]([RS_SWCT_03280](#))

[constr_2056] Consistency of RapidPrototypingScenario with respect to rptSystem and rptArHook references [Within one `RapidPrototypingScenario` all `rptSystem` references shall point to instances in one and only one `System` and if existent all `rptArHook` shall point to instances in one other and only one other `System`.]()

13.3 Usage of atpSplitable for RptContainers on M1

In order to support the later definition of the `RptHooks`, which may require as well the detailed specification `byPassPoints`, the aggregation of `RptContainer` and `RptHook` is `<<atpSplitable>>`.

[TPS_SWCT_02052] Definition of Rapid Prototyping Scenario is splittable [Aggregation of `RptContainer`, `byPassPoint` and `rptHook` using stereotype `<<atpSplitable>>`. By this means it is possible to generally specify the definition the `RptHooks` in a later process step.]([RS_SWCT_03280](#))

Please note that the later specification of `RptHooks` may require additional `byPassPoints` as well to show their relation ship to lower level elements in a component description, such as `VariableAccess` where in contrast the `byPassPoints` may only specified on higher level elements such as `SwComponentPrototypes` in a first step.

13.4 Modifications of the Meta-Model for supporting the RPT scenario

The implementation of the rapid-pro typing scenario implies the definition of *access points* (see table 13.4). To be able to fulfill this role, the *access points* shall be represented by meta-classes derived from [Referrable](#).

Most candidates for becoming *access points* are already inheriting from [Referrable](#) and therefore do not require further treatment (see Figure 13.2). Two meta-classes in this collection, however, are not derived from [Referrable](#):

- [ExternalTriggeringPoint](#)
- [ModeAccessPoint](#)

It is not feasible to fix this issue by simply letting the two meta-classes inherit from [Referrable](#) because this would break the backwards compatibility of the AUTOSAR XML Schema¹. Therefore, a different approach (as sketched in Figure 13.2) has been implemented.

¹Because in this case the [shortName](#) becomes mandatory.

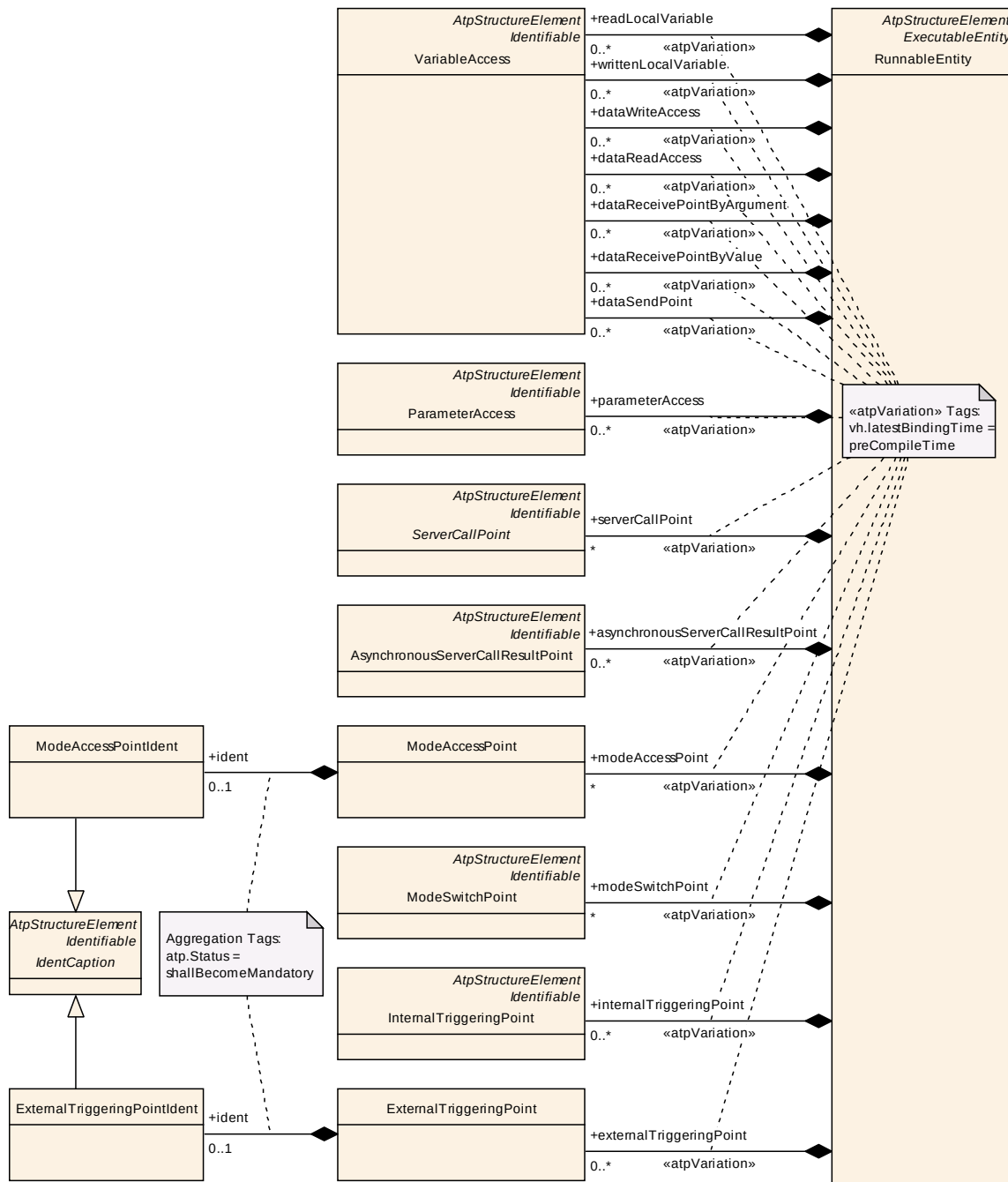


Figure 13.2: Access Points used in the context of the Rapid Prototyping Scenario

A new meta-class `IdentCaption` is created that introduces the capabilities of the meta-class `Identifiable` (that, in turn, inherits from `Referrable`) to its subclasses, `ModeAccessPointIdent` and `ExternalTriggeringPointIdent`.

These, in turn, are **optionally**² aggregated in the role `ident` by `ModeAccessPoint`, resp. in the role `ident` by meta-class `ExternalTriggeringPoint`.

²Again, this is necessary to not break the backwards compatibility

Class	IdentCaption (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::RPTScenario			
Note	This meta-class represents the caption. This allows having some meta classes optionally identifiable.			
Base	ARObject, AtpClassifier, AtpFeature, AtpStructureElement, Identifiable , MultilanguageReferrable, Referrable			
Attribute	Datatype	Mul.	Kind	Note
–	–	–	–	–

Table 13.5: IdentCaption

Class	ModeAccessPointIdent			
Package	M2::AUTOSARTemplates::SWComponentTemplate::RPTScenario			
Note	This meta-class has been created to introduce the ability to become referenced into the meta-class ModeAccessPoint without breaking backwards compatibility.			
Base	ARObject, AtpClassifier, AtpFeature, AtpStructureElement, IdentCaption , Identifiable , MultilanguageReferrable, Referrable			
Attribute	Datatype	Mul.	Kind	Note
–	–	–	–	–

Table 13.6: ModeAccessPointIdent

Class	ExternalTriggeringPointIdent			
Package	M2::AUTOSARTemplates::SWComponentTemplate::RPTScenario			
Note	This meta-class has been created to introduce the ability to become referenced into the meta-class ExternalTriggeringPoint without breaking backwards compatibility.			
Base	ARObject, AtpClassifier, AtpFeature, AtpStructureElement, IdentCaption , Identifiable , MultilanguageReferrable, Referrable			
Attribute	Datatype	Mul.	Kind	Note
–	–	–	–	–

Table 13.7: ExternalTriggeringPointIdent

The following (simplified) listing 13.1 sketches the usage of the meta-class [IdentCaption](#) for the purpose of effectively allowing references to a [ModeAccessPoint](#).

Listing 13.1: Example for the definition of a RPT scenario

```

<AR-PACKAGE>
  <SHORT-NAME>IC_Example</SHORT-NAME>
  <ELEMENTS>
    <APPLICATION-SW-COMPONENT-TYPE>
      <SHORT-NAME>ASCT</SHORT-NAME>
      <INTERNAL-BEHAVIORS>
        <SWC-INTERNAL-BEHAVIOR>
          <SHORT-NAME>IB</SHORT-NAME>
          <RUNNABLES>
            <RUNNABLE-ENTITY>
              <SHORT-NAME>RE</SHORT-NAME>
            <MODE-ACCESS-POINTS>

```

```

        <MODE-ACCESS-POINT>
            <IDENT>
                <SHORT-NAME>ident</SHORT-NAME>
            </IDENT>
        </MODE-ACCESS-POINT>
    </MODE-ACCESS-POINTS>
</RUNNABLE-ENTITY>
</RUNNABLES>
</SWC-INTERNAL-BEHAVIOR>
</INTERNAL-BEHAVIORS>
</APPLICATION-SW-COMPONENT-TYPE>
<COMPOSITION-SW-COMPONENT-TYPE>
    <SHORT-NAME>CSCT</SHORT-NAME>
    <COMPONENTS>
        <SW-COMPONENT-PROTOTYPE>
            <SHORT-NAME>SCP</SHORT-NAME>
            <TYPE-TREF DEST="APPLICATION-SW-COMPONENT-TYPE">/IC_Example
                /ASCT</TYPE-TREF>
        </SW-COMPONENT-PROTOTYPE>
    </COMPONENTS>
</COMPOSITION-SW-COMPONENT-TYPE>
<RAPID-PROTOTYPING-SCENARIO>
    <SHORT-NAME>rptScenario</SHORT-NAME>
    <RPT-CONTAINERS>
        <RPT-CONTAINER>
            <SHORT-NAME>rptContainer</SHORT-NAME>
            <BY-PASS-POINT-IREFS>
                <BY-PASS-POINT-IREF>
                    <CONTEXT-ELEMENT-REF DEST="SW-COMPONENT-PROTOTYPE">/
                        IC_Example/CSCT/SCP</CONTEXT-ELEMENT-REF>
                    <TARGET-REF DEST="MODE-ACCESS-POINT-IDENT">/IC_Example/
                        ASCT/IB/RE/ident</TARGET-REF>
                </BY-PASS-POINT-IREF>
            </BY-PASS-POINT-IREFS>
        </RPT-CONTAINER>
    </RPT-CONTAINERS>
</RAPID-PROTOTYPING-SCENARIO>
</ELEMENTS>
</AR-PACKAGE>

```

A Renamed Meta-Model Elements

A.1 Introduction

In the course of preparing AUTOSAR Release 4.0 some of the existing meta-model elements (as of R 3.x) have been renamed for a better clarity and consistency with respect to other meta-model elements. This chapter provides an overview of the changed meta-model elements in order to allow readers with a background in R3.x specifications to understand changes made by mere renaming.

A.2 Renamed Meta-Model Elements

<i>Old Name</i>	<i>New Name</i>
ApplicationSoftwareComponentType	ApplicationSwComponentType
ArCalprmRef	AutosarParameterRef
ArgumentPrototype	ArgumentDataPrototype
ArrayElement	ApplicationArrayElement
ArrayType	ApplicationArrayDataType
AssemblyConnectorPrototype	AssemblySwConnector
AtomicSoftwareComponentType	AtomicSwComponentType
CalibrationPortAnnotation	ParameterPortAnnotation
CalprmAccess	ParameterAccess
CalprmComponentType	ParameterSwComponentType
CalprmElementPrototype	ParameterDataPrototype
CalprmInterface	ParameterInterface
ComplexDeviceDriverComponentType	ComplexDeviceDriverSwComponentType
ComponentPrototype	SwComponentPrototype
ComponentType	SwComponentType
CompositeType	ApplicationCompositeDataType
CompositionType	CompositionSwComponentType
ConnectorPrototype	SwConnector
DelegationConnectorPrototype	DelegationSwConnector
DataElementPrototype	VariableDataPrototype
DataPrototype	AutosarDataPrototype
Datatype	ApplicationDataType
DirectionKind	ArgumentDirectionEnum
EcuAbstractionComponentType	EcuAbstractionSwComponentType
HandleInvalidType	HandleInvalidEnum
InternalBehavior	SwcInternalBehavior
LimitKind	DataLimitKindEnum
ModeInterface	ModeSwitchInterface
ModeSwitchComSpec	ModeSwitchSenderComSpec
ModeSwitchEvent	SwcModeSwitchEvent
OperationPrototype	ClientServerOperation
PrimitiveType	ApplicationPrimitiveDataType
ProcessingKind	ProcessingKindEnum
RecordElement	ApplicationRecordElement
RecordType	ApplicationRecordDataType
SensorActuatorSoftwareComponentType	SensorActuatorSwComponentType

ServiceComponentType	ServiceSwComponentType
SoftwareComposition	RootSwCompositionPrototype
SwCalprmAxisCommonAxis	SwAxisIndividual
SwCalprmAxisIndividualAxis	SwAxisGrouped
UnqueuedReceiverComSpec	NonqueuedReceiverComSpec
UnqueuedSenderComSpec	NonqueuedSenderComSpec

Table A.1: Renamed meta-model elements

Please note that [InternalBehavior](#) has been moved out of the scope of this specification toward a more general meaning. The original semantics of [InternalBehavior](#) within the scope of this specification is now implemented by [SwcInternalBehavior](#).

B Glossary

Artifact This is a Work Product Definition that provides a description and definition for tangible work product types. Artifacts may be composed of other artifacts ([44]).

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

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

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

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

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

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

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

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

AUTOSAR 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 `<<atpSplittable>>`. 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 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 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`.

C History of Constraints and Specification Items

C.1 Constraint History of this Document according to AUTOSAR R4.0.1

C.1.1 Changed Constraints in R4.0.1

N/A

C.1.2 Added Constraints in R4.0.1

Please note that constraints listed using an bold typeface have been deleted in later versions of the document.

Number	Heading
[constr_1000]	End-to-end protection is limited to sender/receive communication
[constr_1001]	Value of <code>dataId</code> shall be unique
[constr_1002]	End-to-end protection does not support n:1 communication
[constr_1004]	Mapping of <code>ApplicationDataTypes</code>
[constr_1005]	Compatibility of <code>ImplementationDataTypes</code> mapped to the same <code>ApplicationDataType</code>
[constr_1006]	applicable data <code>categorys</code>
[constr_1007]	Allowed attributes of <code>SwDataDefProps</code> for <code>ApplicationDataTypes</code>
[constr_1008]	Applicability of <code>categorys</code> STRUCTURE and ARRAY
[constr_1009]	<code>SwDataDefProps</code> applicable to <code>ImplementationDataTypes</code>
[constr_1010]	If <code>nativeDeclaration</code> does not exist
[constr_1011]	<code>ARMetaClassRolecategoryIdentifiable</code> of <code>SwBaseType</code>
[constr_1012]	Value of <code>category</code> is <code>FIXED_LENGTH</code>
[constr_1013]	Value of <code>category</code> is <code>VARIABLE_LENGTH</code>
[constr_1014]	Supported value encodings for <code>SwBaseType</code>
[constr_1015]	Prioritization of <code>SwDataDefProps</code>
[constr_1016]	<code>invalidValue</code> is restricted
[constr_1017]	Supported combinations of <code>swImplPolicy</code> and <code>swCalibrationAccess</code>
[constr_1018]	<code>measurementPoint</code> shall not be referenced by a <code>VariableAccess</code> aggregated by <code>RunnableEntity</code> in the role <code>dataReadAccess</code>
[constr_1019]	Compatibility of input value and axis
[constr_1020]	<code>ParameterDataPrototype</code> needs to be of compatible data type as referenced in <code>sharedAxisType</code>
[constr_1021]	A <code>CompuMethod</code> shall specify instructions for both directions
[constr_1022]	Limits shall be defined for each direction of <code>CompuMethod</code>
[constr_1023]	Specification of <code>Units</code> in <code>CompuMethods</code>
[constr_1024]	Stepwise definition of <code>CompuMethods</code>
[constr_1025]	Avoid division by zero in rational formula
[constr_1026]	Compatibility of <code>Units</code>
[constr_1027]	Types for record layouts
[constr_1029]	<code>ConstantSpecificationMapping</code> and <code>ConstantSpecification</code>
[constr_1030]	<code>ParameterSwComponentType</code> references <code>ConstantSpecificationMappingSet</code>
[constr_1031]	<code>NvBlockSwComponentType</code> references <code>ConstantSpecificationMappingSet</code>
[constr_1032]	<code>DelegationSwConnector</code> can only connect <code>PortPrototypes</code> of the same kind

[constr_1033]	Communication scenarios for sender/receiver communication
[constr_1035]	Recursive definition of CompositionSwComponentType
[constr_1036]	Connect kinds of PortInterfaces
[constr_1037]	Client may not connect to multiple servers
[constr_1038]	Reference to ApplicationError
[constr_1039]	Relevance of swImplPolicy
[constr_1040]	Conversion of SenderReceiverInterfaces
[constr_1041]	Conversion of ClientServerInterfaces
[constr_1042]	Definition of a linear data scaling
[constr_1043]	PortInterface vs. ComSpec
[constr_1044]	Applicability of DataFilter
[constr_1045]	Supported value encodings for SwBaseType in the context of PortInterfaces
[constr_1046]	Applicability of [constr_1045]
[constr_1047]	Compatibility of ApplicationPrimitiveDataTypes
[constr_1048]	Compatibility of ApplicationRecordDataTypes
[constr_1049]	Compatibility of ApplicationArrayDataTypes
[constr_1050]	Compatibility of ImplementationDataTypes
[constr_1051]	Compatibility of SwDataDefProps
[constr_1052]	Compatibility of Units
[constr_1053]	Compatibility of PhysicalDimensions
[constr_1054]	No DataConstr available at the provider
[constr_1055]	ImplementationDataType has category VALUE
[constr_1056]	ImplementationDataType has category TYPE_REFERENCE
[constr_1057]	ImplementationDataType has category DATA_REFERENCE
[constr_1058]	ImplementationDataType has category FUNCTION_REFERENCE
[constr_1059]	Compatibility of data types with category VALUE
[constr_1060]	Compatibility of data types with category ARRAY , VAL_BLK , or STRING
[constr_1061]	Compatibility of data types with category STRUCTURE
[constr_1062]	Compatibility of data types with category BIT
[constr_1063]	Compatibility of data types with category BOOLEAN
[constr_1064]	Compatibility of data types with category COM_AXIS , RES_AXIS , CURVE or MAP
[constr_1066]	ApplicationDataType is or is not compatible to specific ImplementationDataType
[constr_1067]	ApplicationDataType is or is not compatible to specific ImplementationDataType
[constr_1068]	Compatibility of VariableDataPrototypes or ParameterDataPrototypes typed by primitive data types
[constr_1069]	Compatibility of PortPrototypes of different DataInterfaces in the context of AssemblySwConnectors
[constr_1070]	Compatibility of PortPrototypes of different DataInterfaces in the context of DelegationSwConnectors
[constr_1071]	compatibility of compatibility of ParameterDataPrototype and VariableDataPrototype
[constr_1072]	Compatibility of ModeSwitchInterfaces in the context of an AssemblySwConnector
[constr_1073]	Compatibility of ModeSwitchInterfaces in the context of an DelegationSwConnector
[constr_1074]	Compatibility of ModeDeclarationGroupPrototypes
[constr_1075]	Compatibility of ModeDeclarationGroups
[constr_1076]	Compatibility of ArgumentDataPrototypes
[constr_1077]	Compatibility of ApplicationErrors
[constr_1078]	Compatibility of ClientServerOperations

[constr_1079]	Compatibility of ClientServerInterfaces in the context of an AssemblySwConnector
[constr_1080]	Compatibility of ClientServerInterfaces in the context of an DelegationSwConnector
[constr_1081]	Compatibility of TriggerInterfaces in the context of an AssemblySwConnector
[constr_1082]	Compatibility of TriggerInterfaces in the context of an DelegationSwConnector
[constr_1083]	Compatibility of Triggers
[constr_1084]	delegation of an provided outer PortPrototype
[constr_1085]	Compatibility in the case of a flat ECU extract
[constr_1086]	SwConnector between two specific PortPrototypes
[constr_1087]	AssemblySwConnector inside CompositionSwComponentType
[constr_1088]	DelegationSwConnector inside CompositionSwComponentType
[constr_1090]	WaitPoint and RunnableEntity
[constr_1091]	RTEEvents that can unblock a WaitPoint
[constr_1092]	ParameterSwComponentType
[constr_1093]	Definition of textual strings
[constr_1094]	Usage of symbol of RunnableEntity
[constr_1095]	Values of nDataSets vs. reliability
[constr_1096]	SwcModeSwitchEvent and WaitPoint
[constr_1097]	RunnableEntity that has a WaitPoint
[constr_1098]	Mode switch and mode disabling
[constr_1099]	Data type of inter-runnable variables
[constr_1100]	Unconnected RPortPrototype typed by a DataInterface
[constr_1101]	Mode-related communication
[constr_1102]	ApplicationError in the scope of one SwComponentType
[constr_1103]	NonqueuedReceiverComSpec and enableUpdate
[constr_1104]	Trigger sink and trigger source
[constr_1105]	Value of arraySize
[constr_1106]	Structure shall have at least one element
[constr_1107]	Union shall have at least one element
[constr_1108]	Value of ApplicationError.errorCode
[constr_1109]	Mapping of SwComponentPrototypes typed by a SensorActuatorSwComponentType
[constr_1110]	Value of category in EndToEndDescription
[constr_1111]	Constraints of dataId in PROFILE_01
[constr_1112]	Constraints of dataIdMode in PROFILE_01
[constr_1113]	Existence of attributes in PROFILE_01
[constr_1114]	Constraints of crcOffset in PROFILE_01
[constr_1115]	Constraints of counterOffset in PROFILE_01
[constr_1116]	Constraints of dataLength in PROFILE_01
[constr_1117]	Constraints of maxDeltaCounterInit in PROFILE_01
[constr_1118]	Existence of attributes in PROFILE_02
[constr_1119]	Constraints of dataLength in PROFILE_02
[constr_1120]	Constraints of dataId in PROFILE_02
[constr_1121]	Constraints of maxDeltaCounterInit in PROFILE_02
[constr_1122]	Existence of attributes in PROFILE_03
[constr_1123]	Constraints of dataLength in PROFILE_03
[constr_1124]	Constraints of dataId in PROFILE_03
[constr_1125]	Constraints of maxDeltaCounterInit in PROFILE_03
[constr_1126]	Compatibility of DataConstrs
[constr_2000]	Compatibility of ClientServerOperations triggering the same RunnableEntity

[constr_2001]	Initial value for a specific implicitInterRunnableVariable or explicitInterRunnableVariable
[constr_2002]	Referenced VariableDataPrototype from AutosarVariableRef of VariableAccess in role dataReadAccess
[constr_2003]	Referenced VariableDataPrototype from AutosarVariableRef of VariableAccess in role dataWriteAccess
[constr_2004]	Referenced VariableDataPrototype from AutosarVariableRef of VariableAccess in role dataSendPoint
[constr_2005]	Referenced VariableDataPrototype from AutosarVariableRef of VariableAccess in role dataReceivePointByValue or dataReceivePointByArgument
[constr_2006]	Number of AsynchronousServerCallResultPoint referencing to one AsynchronousServerCallPoint
[constr_2007]	Consistency of typeDefinition attribute
[constr_2009]	Supported kinds of ports of a NvBlockSwComponentType
[constr_2010]	Connections between SwComponentPrototypes of type NvBlockSwComponentType
[constr_2011]	Connections between SwComponentPrototypes typed by NvBlockSwComponentType and SwComponentPrototypes typed by other AtomicSwComponentTypes
[constr_2012]	Compatibility of ImplementationDataTypes used for ramBlock and romBlock
[constr_2013]	Compatibility of ImplementationDataTypes for NvBlockDataMapping
[constr_2014]	Limitation of RoleBasedPortAssignment.role in NvBlockDescriptors
[constr_2015]	Limitation of SwcInternalBehavior of a NvBlockSwComponentType
[constr_2016]	Connections between SwComponentPrototypes of type ServiceProxySwComponentType
[constr_2017]	Ports of ServiceProxySwComponentTypes
[constr_2018]	Supported remote communication of a ServiceProxySwComponentType
[constr_2019]	ServiceSwComponentType shall have service ports only
[constr_2020]	dataReadAccess can not be used for queued communication
[constr_2021]	WaitPoint referencing a DataReceivedEvent can not be used for non-queued communication
[constr_2022]	Mutually exclusive use of SynchronousServerCallPoints and AsynchronousServerCallPoints
[constr_2023]	Consistency of timeout values
[constr_2024]	enableTakeAddress is restricted to single instantiation
[constr_2025]	Uniqueness of symbol attributes
[constr_2026]	Referenced VariableDataPrototype from AutosarVariableRef of VariableAccess in role writtenLocalVariable and readLocalVariable
[constr_2027]	SwcServiceDependency shall be defined for service ports only
[constr_2028]	staticMemory is restricted to single instantiation
[constr_2029]	shortName of constantMemory and staticMemory
[constr_2030]	AsynchronousServerCallResultPoint combined with WaitPoint shall belong to the same RunnableEntity
[constr_2031]	Period of TimingEvent shall be greater than 0
[constr_2032]	transmissionAcknowledge requires a DataSendCompletedEvent
[constr_2033]	Timeout of DataSendCompletedEvent
[constr_2500]	PortInterfaces shall be of same kind
[constr_2526]	PortInterfaces need to be compatible to the blueprints
[constr_2527]	Blueprints shall live in package of a proper category
[constr_2528]	PortPrototypes shall not refer to blueprints of PortInterfaces
[constr_2529]	Blueprints of ports and interfaces shall be compatible

[constr_2533]	Iteration along output axis is only supported for VALUE and VAL_BLK
[constr_4000]	Local communication of mode switches
[constr_4001]	Content of ModeRequestTypeMap
[constr_4002]	Unambiguous mapping of modes to data types
[constr_4003]	Semantics of SwcModeSwitchEvent
[constr_4004]	Context of SenderReceiverAnnotation
[constr_4005]	Context of ClientServerAnnotation
[constr_4006]	Context of ParameterPortAnnotation
[constr_4007]	Context of ModePortAnnotation
[constr_4008]	Context of TriggerPortAnnotation
[constr_4009]	Context of NvDataPortAnnotation
[constr_4010]	Context of DelegatedPortAnnotation
[constr_4011]	ComSpec and ModeSwitchedAckEvent
[constr_4012]	Timeout of ModeSwitchedAckEvent
[constr_4035]	ValueSpecification shall fit into data type

Table C.1: Added Constraints in R4.0.1

C.1.3 Deleted Constraints

N/A

C.2 Constraint History of this Document according to AUTOSAR R4.0.2

C.2.1 Changed Constraints in R4.0.2

Please note that constraints listed using an bold typeface have been deleted in later versions of the document.

Number	Heading
[constr_1007]	Allowed attributes of SwDataDefProps for ApplicationDataTypes
[constr_1061]	Compatibility of data types with category STRUCTURE
[constr_2001]	Initial value for a specific implicitInterRunnableVariable or explicitInterRunnableVariable

Table C.2: Changed Constraints in R4.0.2

C.2.2 Added Constraints in R4.0.2

Please note that constraints listed using an bold typeface have been deleted in later versions of the document.

Number	Heading
[constr_1127]	ServiceSwComponentType shall not have ServiceNeeds

[constr_1128]	Queue length of ClientServerOperations associated with the same RunnableEntity
[constr_1129]	swImplPolicy and NonqueuedReceiverComSpec
[constr_1130]	swImplPolicy and NonqueuedReceiverComSpec
[constr_1131]	swImplPolicy and NonqueuedSenderComSpec
[constr_1132]	swImplPolicy and NonqueuedSenderComSpec
[constr_1133]	Identical CompuScale Symbolic Names shall have the same range
[constr_1134]	Allowed structure of TEXTTABLE
[constr_1135]	Limit of vt in BITFIELD_TEXTTABLE
[constr_1136]	Compatibility of introduction of blueprint and blueprinted element
[constr_1137]	Applicability of ParameterInterface
[constr_1138]	assignedPort and DiagEventDebounceMonitorInternal
[constr_1139]	assignedPort of DiagEventDebounceMonitorInternal shall refer to an RPortPrototype
[constr_2034]	SwAddrMethod referenced by RunnableEntity s or BswSchedulableEntity s
[constr_2035]	swImplPolicy for VariableDataPrototype in SenderReceiverInterface
[constr_2036]	swImplPolicy for VariableDataPrototype in NvDataInterface
[constr_2037]	swImplPolicy for VariableDataPrototype in the role ramBlock
[constr_2038]	swImplPolicy for VariableDataPrototype in the role implicitInter-RunnableVariable
[constr_2039]	swImplPolicy for VariableDataPrototype in the role explicitInter-RunnableVariable
[constr_2040]	swImplPolicy for VariableDataPrototype in the role arTypedPerInstance-Memory
[constr_2041]	swImplPolicy for VariableDataPrototype in the role staticMemory
[constr_2042]	swImplPolicy for ParameterDataPrototype in ParameterInterface
[constr_2043]	swImplPolicy for ParameterDataPrototype in the role staticMemory
[constr_2044]	swImplPolicy for ParameterDataPrototype in the role sharedParameter
[constr_2045]	swImplPolicy for ParameterDataPrototype in the role perInstanceParameter
[constr_2046]	swImplPolicy for ParameterDataPrototype in the role constantMemory
[constr_2047]	swImplPolicy for ArgumentDataPrototype
[constr_2048]	swImplPolicy for SwServiceArg
[constr_2535]	Target of an autosarParameter in AutosarParameterRef shall refer to a parameter
[constr_2536]	Target of an autosarVariable in AutosarVariableRef shall refer to a variable

Table C.3: Added Constraints in R4.0.2

C.2.3 Deleted Constraints in R4.0.2

Number	Heading
[constr_1099]	Data type of inter-runnable variables

Table C.4: Deleted Constraints in R4.0.2

C.3 Constraint History of this Document according to AUTOSAR R4.0.3

C.3.1 Changed Constraints in R4.0.3

Please note that constraints listed using an bold typeface have been deleted in later versions of the document.

Number	Heading
[constr_1006]	applicable data <code>categorys</code>
[constr_1009]	<code>SwDataDefProps</code> applicable to <code>ImplementationDataTypes</code>
[constr_1014]	Supported value encodings for <code>SwBaseType</code>
[constr_1015]	Prioritization of <code>SwDataDefProps</code>
[constr_1043]	<code>PortInterface</code> vs. <code>ComSpec</code>
[constr_1051]	Compatibility of <code>SwDataDefProps</code>
[constr_1053]	Compatibility of <code>PhysicalDimensions</code>
[constr_1063]	Compatibility of data types with <code>category</code> <code>BOOLEAN</code>
[constr_1110]	Value of <code>category</code> in <code>EndToEndDescription</code>
[constr_1113]	Existence of attributes in <code>PROFILE_01</code>
[constr_1118]	Existence of attributes in <code>PROFILE_02</code>
[constr_1134]	Allowed structure of <code>TEXTTABLE</code>
[constr_2000]	Compatibility of <code>ClientServerOperations</code> triggering the same <code>RunnableEntity</code>
[constr_2027]	<code>SwcServiceDependency</code> shall be defined for service ports only

Table C.5: Changed Constraints in R4.0.3

C.3.2 Added Constraints in R4.0.3

Please note that constraints listed using an bold typeface have been deleted in later versions of the document.

Number	Heading
[constr_1140]	Combination of <code>invalidValue</code> with the attribute <code>handleInvalid</code>
[constr_1141]	Applicability of the <code>scope</code> attribute
[constr_1142]	<code>category</code> of <code>CompuMethod</code> shall not be extended
[constr_1143]	<code>category</code> of <code>AutosarDataType</code> shall not be extended
[constr_1144]	<code>SensorActuatorSwComponentType</code> , <code>EcuAbstractionSwComponentType</code> , and <code>ComplexDeviceDriverSwComponentType</code> may only reference a <code>HwType</code>
[constr_1145]	Finding the symbol for the representation of a <code>CompuScale</code> in C code
[constr_1146]	Applicability of a symbol for a <code>CompuScale</code> in C code
[constr_1147]	Standardized values for the attribute <code>category</code> of meta-class <code>PortGroup</code>
[constr_1148]	<code>PortInterfaces</code> of <code>PortPrototypes</code> used to connect to <code>NvBlockSwComponentTypes</code>
[constr_1149]	<code>PortPrototypes</code> used for NV data management
[constr_1150]	Usage of <code>valueType</code> for <code>PortDefinedArgumentValue</code>
[constr_1151]	Applicability of <code>PortInterfaceMapping</code>
[constr_1151]	<code>category</code> of <code>ApplicationArrayElement</code> and <code>AutosarDataType</code> referenced in the role <code>type</code> shall be kept in sync
[constr_1153]	Applicability of compatibility requirements for <code>CompuScales</code>
[constr_1154]	Compatibility of <code>CompuScales</code> for sender-receiver communication and similar use cases

[constr_1155]	Compatibility of <code>CompuScales</code> for client-server communication
[constr_1156]	Relevance of “names” of <code>CompuScales</code>
[constr_1157]	Applicability of constraints of <code>CompuScales</code>
[constr_1158]	Applicable <code>category</code> s for attribute <code>compuMethod</code>
[constr_1159]	Consistency of <code>VariableAndParameterInterfaceMapping</code> with respect to the referenced <code>DataInterfaces</code>
[constr_1160]	Size of Compound Primitive Data Type is variant
[constr_1161]	Applicability of the <code>index</code> attribute of <code>Ref</code>
[constr_1162]	Compatibility of <code>SwRecordLayouts</code>
[constr_1163]	Compatibility of <code>CompuMethods</code>
[constr_1164]	Number of <code>arguments</code> owned by a <code>RunnableEntity</code>
[constr_1165]	Applicability of <code>RunnableEntityArgument</code>
[constr_1166]	Restrictions of <code>ModeRequestTypeMap</code>
[constr_1167]	<code>ImplementationDataTypes</code> used as <code>ModeRequestTypeMap.implementationDataType</code>
[constr_1168]	Compatibility of <code>ImplementationDataTypes</code> used used in the <code>ModeRequestTypeMap</code>
[constr_1169]	Allowed values for <code>Trigger.swImplPolicy</code>
[constr_1170]	Interpretation of attribute <code>maxDeltaCounterInit</code> owned by <code>EndToEndDescription</code>
[constr_1171]	Interpretation of attribute <code>maxDeltaCounterInit</code> of <code>EndToEndDescription</code>
[constr_1172]	Allowed values of <code>SwCalibrationAccessEnum</code> for <code>ModeDeclarationGroupPrototype</code>
[constr_1173]	Applicability of <code>AutosarParameterRef</code> referencing a <code>VariableDataPrototype</code>
[constr_1174]	<code>PortInterfaces</code> used in the context of <code>CompositionSwComponentTypes</code> cannot refer to AUTOSAR services
[constr_1175]	Depending on its <code>category</code> , <code>CompuMethod</code> shall refer to a unit
[constr_1176]	Compatibility of <code>CompuScales</code> of <code>category</code> <code>LINEAR</code> and <code>RAT_FUNC</code>
[constr_1177]	Allowed <code>category</code> for <code>SwPointerTargetProps</code>
[constr_1178]	Existence of attributes of <code>SwDataDefProps</code> in the context of <code>ImplementationDataType</code>
[constr_1179]	Existence of <code>ModeDeclaration.value</code> within a <code>ModeDeclarationGroup</code>
[constr_1180]	Existence of <code>ModeDeclarationGroup.onTransitionValue</code>
[constr_1181]	Numerical values used in <code>ModeDeclaration.value</code> and <code>ModeDeclarationGroup.onTransitionValue</code>
[constr_1182]	Allowed values for <code>InternalTriggeringPoint.swImplPolicy</code>
[constr_1183]	<code>EndToEndProtectionVariablePrototypes</code> aggregated by <code>EndToEndProtection</code>
[constr_1184]	Consistency of <code>rootDataPrototype</code> and <code>base</code> in the context of <code>ApplicationCompositeElementInPortInterfaceInstanceRef</code>
[constr_1185]	Consistency of data types in the context of <code>ApplicationCompositeElementInPortInterfaceInstanceRef</code>
[constr_1186]	Consistency of data types in the context of <code>ArVariableInImplementationDataInstanceRef</code>
[constr_1187]	Compatibility of <code>VariableDataPrototypes</code> or <code>ParameterDataPrototypes</code> typed by composite data types
[constr_1188]	Existence of <code>ReceiverComSpec.replaceWith</code>
[constr_1189]	Allowed targets of <code>externalReplacement</code>
[constr_1190]	Only one mapping for composite to primitive use case
[constr_2049]	Different <code>ModeDeclarationGroups</code> shall have different <code>shortNames</code> .
[constr_2050]	Mandatory information of a <code>SwAxisCont</code>
[constr_2051]	Mandatory information of a <code>SwValueCont</code>

[constr_2052]	Values of <code>swArraySize</code> and the number of values provided by <code>swValuesPhys</code> shall be consistent.
[constr_2053]	Consistency between <code>role</code> <code>IUMPRNumerator</code> and <code>ObdRatioServiceNeeds.connectionType</code>
[constr_2544]	Limits need to be consistent
[constr_2545]	<code>invalidValue</code> shall fit in the specified ranges
[constr_2548]	Data constraint of value axis shall match
[constr_2549]	Units of input axis shall be consistent
[constr_2550]	Units of value axis shall be consistent
[constr_2551]	<code>SwCalprmAxis.baseType</code> shall be ignored
[constr_2561]	Application of <code>DataConstrRule.constrLevel</code>

Table C.6: Added Constraints in R4.0.3

C.3.3 Added Specification Items in R4.0.3

Please note that specification items listed using an bold typeface have been deleted in later versions of the document.

Number	Heading
[TPS_SWCT_01000]	Usage of attribute <code>symbol</code> of the <code>symbolProps</code>
[TPS_SWCT_01001]	Prefix symbols generated for the <code>RunnableEntity</code>
[TPS_SWCT_01002]	<code>SwComponentTypes</code> may only interact by means of their <code>PortPrototypes</code>
[TPS_SWCT_01003]	Inconsistencies regarding the value of <code>serviceKind</code> and the actual implementation of the <code>PortInterface</code>
[TPS_SWCT_01004]	Default value if <code>serviceKind</code> is not defined
[TPS_SWCT_01005]	Usage of <code>SwcServiceDependency</code> s for vendor-specific services
[TPS_SWCT_01006]	<code>arraySize</code> of <code>ImplementationDataType</code> shall be used to define the size of the array
[TPS_SWCT_01007]	Semantics of array index
[TPS_SWCT_01008]	Definition of positive integer values that are directly taken over by the RTE generator for creating the programmatic representations of the <code>ModeDeclaration</code>
[TPS_SWCT_01009]	The numerical values used to define the values of <code>ModeDeclaration.value</code> and <code>ModeDeclarationGroup.onTransitionValue</code> can be arbitrarily defined
[TPS_SWCT_01010]	<code>category</code> s for the definition of a <code>ModeDeclarationGroup</code>
[TPS_SWCT_01011]	Default <code>category</code> of a <code>ModeDeclarationGroup</code>
[TPS_SWCT_01012]	<code>AtomicSwComponentType</code> reads the current ECU mode (fixed variant)
[TPS_SWCT_01013]	<code>AtomicSwComponentType</code> shall keep the ECU alive (fixed variant)
[TPS_SWCT_01014]	<code>AtomicSwComponentType</code> wants to select a shutdown target (fixed variant)
[TPS_SWCT_01015]	<code>AtomicSwComponentType</code> wants to select a boot target (fixed variant)
[TPS_SWCT_01016]	<code>AtomicSwComponentType</code> wants to select a shutdown target (flexible variant)
[TPS_SWCT_01017]	<code>AtomicSwComponentType</code> wants to select a boot target (flexible variant)
[TPS_SWCT_01018]	<code>AtomicSwComponentType</code> wants to use an alarm clock (flexible variant)
[TPS_SWCT_01019]	<code>AtomicSwComponentType</code> reads the current ComM mode
[TPS_SWCT_01020]	<code>AtomicSwComponentType</code> requests a ComM mode. It may also check later whether the requested ComM mode has become effective
[TPS_SWCT_01021]	<code>AtomicSwComponentType</code> acts as a mode manager that influences the ECU state
[TPS_SWCT_01022]	Queued processing of internal trigger

[TPS_SWCT_01023]	Mapping of elements of composite data types
[TPS_SWCT_01024]	Combination of ApplicationCompositeDataType and nested ImplementationDataType
[TPS_SWCT_01025]	The role of PortPrototypes in the AUTOSAR architecture
[TPS_SWCT_01026]	The role of PortInterfaces in the AUTOSAR architecture
[TPS_SWCT_01027]	Different flavors of PortInterfaces
[TPS_SWCT_01028]	AtomicSwComponentType implements a Diagnostic Monitor
[TPS_SWCT_01029]	AtomicSwComponentType implements a Diagnostic Monitor
[TPS_SWCT_01030]	RunnableEntity
[TPS_SWCT_01031]	ExclusiveArea
[TPS_SWCT_01032]	CompositionSwComponentType
[TPS_SWCT_01033]	Nested definition of CompositionSwComponentTypes
[TPS_SWCT_01034]	CompositionSwComponentTypes do not have any binary footprint
[TPS_SWCT_01035]	CompositionSwComponentType aggregates SwComponentPrototypes
[TPS_SWCT_01036]	SwComponentPrototype implements a specific role
[TPS_SWCT_01037]	arbitrary numbers of SwComponentPrototypes can be created
[TPS_SWCT_01038]	Support for Variant Handling in the in Software Component Template
[TPS_SWCT_01039]	Purpose of variant handling
[TPS_SWCT_01040]	SwConnector exists depending on a <i>PostBuild</i> condition
[TPS_SWCT_01041]	API functions of not existing SwConnector are still part of the software-component's implementation
[TPS_SWCT_01042]	Four types of locations in the meta-model which may exhibit variability
[TPS_SWCT_01043]	ApplicationSwComponentTypes are independent from actual ECU Hardware
[TPS_SWCT_01044]	ServiceNeeds
[TPS_SWCT_01045]	Actual values of ECU configuration parameters fulfill the requirements given by the ServiceNeeds
[TPS_SWCT_01046]	ServiceNeeds are defined in the scope of the SwcInternalBehavior
[TPS_SWCT_01047]	Reference from the software representation of a sensor/actuator to the actual hardware element
[TPS_SWCT_01048]	SensorActuatorSwComponentType may use the I/O hardware abstraction directly
[TPS_SWCT_01049]	Two ways to use the ExclusiveAreas
[TPS_SWCT_01050]	RunnableEntity always runs inside an ExclusiveArea
[TPS_SWCT_01051]	RunnableEntity explicitly enters and leaves a specific ExclusiveArea
[TPS_SWCT_01052]	Inter-runnable variable
[TPS_SWCT_01053]	Relationship of interchanged data with RunnableEntities
[TPS_SWCT_01054]	Semantics of the explicitInterRunnableVariable
[TPS_SWCT_01055]	Semantics of implicitInterRunnableVariable
[TPS_SWCT_01056]	Physical dimension
[TPS_SWCT_01057]	Unit references one physical dimension
[TPS_SWCT_01058]	UnitGroup
[TPS_SWCT_01059]	Exponent for each of the seven fundamental dimensions
[TPS_SWCT_01060]	Negative exponents
[TPS_SWCT_01061]	Conversion of units
[TPS_SWCT_01062]	Documentation of software-components
[TPS_SWCT_01063]	PortGroup
[TPS_SWCT_01064]	PortGroups have to be defined on the VFB level
[TPS_SWCT_01065]	PortPrototype may belong to more than one PortGroups
[TPS_SWCT_01066]	PortGroups can be associated with certain ServiceNeeds
[TPS_SWCT_01067]	Initial mode
[TPS_SWCT_01068]	Units can be grouped with the help of UnitGroup

[TPS_SWCT_01069]	DataInterface is defined as abstract base class
[TPS_SWCT_01070]	PortInterface acts as a <i>type</i> for a PortPrototype
[TPS_SWCT_01071]	ModeDeclaration
[TPS_SWCT_01072]	ApplicationDataType and ImplementationDataType
[TPS_SWCT_01073]	Composite ApplicationDataType
[TPS_SWCT_01074]	Composite ImplementationDataType
[TPS_SWCT_01075]	SwcInternalBehavior
[TPS_SWCT_01076]	Number of elements of a specific ApplicationArrayDataType might vary at run-time
[TPS_SWCT_01077]	Configure the response to mode changes
[TPS_SWCT_01078]	Configurable array size
[TPS_SWCT_01079]	SwConnector
[TPS_SWCT_01080]	Delegation ports
[TPS_SWCT_01081]	Implications of being a delegation port
[TPS_SWCT_01082]	AssemblySwConnector
[TPS_SWCT_01083]	DelegationSwConnector
[TPS_SWCT_01084]	Outer PortPrototype is referenced by multiple DelegationSwConnectors
[TPS_SWCT_01085]	Variation on the behavior level
[TPS_SWCT_01086]	Request mode change
[TPS_SWCT_01087]	Propagation of mode information
[TPS_SWCT_01088]	ComSpecs defined by CompositionSwComponentTypes
[TPS_SWCT_01089]	end-to-end communication protection
[TPS_SWCT_01090]	EndToEndProtection
[TPS_SWCT_01091]	Two cases for end-to-end protection
[TPS_SWCT_01092]	EndToEndProtectionSet
[TPS_SWCT_01093]	Definition of end-to-end protection is splittable
[TPS_SWCT_01094]	<i>category</i> of EndToEndDescription
[TPS_SWCT_01095]	<i>category</i> set to NONE
[TPS_SWCT_01096]	PortGroup
[TPS_SWCT_01097]	CompositionSwComponentType cannot have RunnableEntitys
[TPS_SWCT_01098]	Only AtomicSwComponentType can have RunnableEntitys
[TPS_SWCT_01099]	PortInterfaceMapping
[TPS_SWCT_01100]	Precedence of PortInterfaceMapping
[TPS_SWCT_01101]	Unmapped elements of PortInterfaces
[TPS_SWCT_01102]	VariableAndParameterInterfaceMapping
[TPS_SWCT_01103]	Mapping between different kinds of PortInterfaces
[TPS_SWCT_01104]	Possible mappings are restricted by the swImplPolicy
[TPS_SWCT_01105]	ClientServerInterfaceMapping
[TPS_SWCT_01106]	ClientServerOperation
[TPS_SWCT_01107]	swMinAxisPoints and swMaxAxisPoints represent variation points
[TPS_SWCT_01108]	Added value of an AtomicSwComponentType
[TPS_SWCT_01109]	Adding the SwcInternalBehavior in a later process step
[TPS_SWCT_01110]	Symbolic name of a software-component
[TPS_SWCT_01111]	PortPrototypes need an additional model artifact, the PortInterface
[TPS_SWCT_01112]	PortPrototypes are either <i>require-</i> or <i>provide-</i> ports.
[TPS_SWCT_01113]	Connecting two PortPrototypes
[TPS_SWCT_01114]	SenderReceiverInterface
[TPS_SWCT_01115]	InvalidationPolicy
[TPS_SWCT_01116]	swImplPolicy
[TPS_SWCT_01117]	Communication patterns for sender-receiver communication
[TPS_SWCT_01118]	ClientServerInterface
[TPS_SWCT_01119]	Direction of ArgumentDataPrototypes

[TPS_SWCT_01120]	Client needs to provide ArgumentDataPrototypes
[TPS_SWCT_01121]	Pass correct data type
[TPS_SWCT_01122]	Synchronous call of ClientServerOperation
[TPS_SWCT_01123]	No default values for ArgumentDataPrototypes
[TPS_SWCT_01124]	Definition of ArgumentDataPrototypes within the context of a ClientServerOperation is ordered
[TPS_SWCT_01125]	serverArgumentImplPolicy
[TPS_SWCT_01126]	Access to partial networking via BswM
[TPS_SWCT_01127]	Byte array with variable size
[TPS_SWCT_01128]	SwRecordLayout needed
[TPS_SWCT_01129]	Express diagnostic capabilities
[TPS_SWCT_01130]	Measurement and calibration access to model elements is defined by swCalibrationAccess
[TPS_SWCT_01131]	AtomicSwComponentType accepts a request to restart an entire function
[TPS_SWCT_01132]	AtomicSwComponentType provides information about operating cycles
[TPS_SWCT_01133]	AtomicSwComponentType provides information about aging cycles
[TPS_SWCT_01134]	AtomicSwComponentType enables storage of DTCs in general
[TPS_SWCT_01135]	AtomicSwComponentType enables storage of subsequent DTCs
[TPS_SWCT_01136]	AtomicSwComponentType retrieves information from the fault storage
[TPS_SWCT_01137]	Dem provides information that the fault storage overflows
[TPS_SWCT_01138]	AtomicSwComponentType suppresses the storage of DTCs within the Dem
[TPS_SWCT_01139]	AtomicSwComponentType informs the Dem that the PTO is active
[TPS_SWCT_01140]	AtomicSwComponentType needs information about specific DTC without being a diagnostic monitor
[TPS_SWCT_01141]	AtomicSwComponentType may have RPortPrototypes typed by an NvDataInterface
[TPS_SWCT_01142]	non-volatile data are provided by a specialized AtomicSwComponentType
[TPS_SWCT_01143]	Non-volatile data represented by an NvBlockComponent can be read and written
[TPS_SWCT_01144]	NvBlockDescriptor specifies the properties of exactly one NvBlock
[TPS_SWCT_01145]	ramBlock and the romBlock are described by a VariableDataPrototype and a ParameterDataPrototype
[TPS_SWCT_01146]	romBlock is optional
[TPS_SWCT_01147]	No romBlock is configured
[TPS_SWCT_01148]	NvBlockDataMapping
[TPS_SWCT_01149]	RoleBasedPortAssignment of NvBlockDescriptor
[TPS_SWCT_01150]	InternalBehavior of a NvBlockSwComponentType
[TPS_SWCT_01151]	RunnableEntity s do not have further attributes
[TPS_SWCT_01152]	InternalBehavior does not have further attributes
[TPS_SWCT_01153]	IncludedModeDeclarationGroupSet
[TPS_SWCT_01154]	Attribute prefix of IncludedModeDeclarationGroupSet
[TPS_SWCT_01155]	IncludedDataSet
[TPS_SWCT_01156]	Required if the AutosarDataType is not used for any DataPrototype
[TPS_SWCT_01157]	Attribute literalPrefix of IncludedDataSet
[TPS_SWCT_01158]	Three cases for PortInterfaceMapping
[TPS_SWCT_01159]	Mapping is described separately from the SwConnector as reusable ARElement
[TPS_SWCT_01160]	ModeInterfaceMapping
[TPS_SWCT_01161]	TriggerInterfaceMapping
[TPS_SWCT_01162]	Conditional existence of TextTableMapping
[TPS_SWCT_01163]	Conversion from firstValue to secondValue
[TPS_SWCT_01164]	Conversion from secondValue to firstValue
[TPS_SWCT_01165]	Invertible mapping

[TPS_SWCT_01166]	Non-invertible mapping
[TPS_SWCT_01167]	Validity of ModeInterfaceMapping
[TPS_SWCT_01168]	Linear conversion factor can be calculated
[TPS_SWCT_01169]	Support for partial networking
[TPS_SWCT_01170]	Purpose of Virtual Function Cluster
[TPS_SWCT_01171]	Purpose of a control port
[TPS_SWCT_01172]	Requesting and releasing partial networks
[TPS_SWCT_01173]	Control port shall not become a part of the PortGroup
[TPS_SWCT_01174]	Status port shall not become a member of the PortGroup
[TPS_SWCT_01175]	Actively query the status of a partial network
[TPS_SWCT_01176]	last-is-best semantics for sender-receiver communication
[TPS_SWCT_01177]	Assignment of constant values
[TPS_SWCT_01178]	Specialized subclasses of ValueSpecification
[TPS_SWCT_01179]	Compound Primitive Data Type
[TPS_SWCT_01180]	Maximum possible size of Compound Primitive Data Type
[TPS_SWCT_01181]	Bound model specifies a primitive which is smaller than the maximum defined by the range of the involved SwSystemconst
[TPS_SWCT_01182]	Conceptual levels for the definition of initial values
[TPS_SWCT_01183]	Actual value of an <code>initValue</code> shall be interpreted according to the Autosar-DataType
[TPS_SWCT_01184]	ApplicationPrimitiveDataTypes with <code>category</code> VALUE
[TPS_SWCT_01185]	<code>initValues</code> for Compound Primitive Data Types
[TPS_SWCT_01186]	ConstantSpecificationMapping
[TPS_SWCT_01187]	ConstantSpecificationMappingSet referenced by the InternalBehavior
[TPS_SWCT_01188]	Definition of calibration data sets through RTE-generator and compiler
[TPS_SWCT_01189]	DataTypeMap
[TPS_SWCT_01190]	ModeRequestTypeMap
[TPS_SWCT_01191]	mapped ApplicationDataType and ImplementationDataType shall be compatible
[TPS_SWCT_01192]	Meta-classes that have an association to a DataTypeMappingSet
[TPS_SWCT_01193]	Mappings between application and implementation types do not necessarily have to form a 1:1 relation
[TPS_SWCT_01194]	Symbolic name of an ImplementationDataType
[TPS_SWCT_01195]	Mapping of composite element to primitive DataPrototype
[TPS_SWCT_01196]	Semantics of an external trigger event communication
[TPS_SWCT_01197]	TriggerInterface
[TPS_SWCT_01198]	Period for periodic triggering
[TPS_SWCT_01199]	Queued processing of Triggers
[TPS_SWCT_01200]	ModeDeclarationGroupPrototype per ModeSwitchInterface
[TPS_SWCT_01201]	CompositionSwComponentType requires and provides the modes that are required or provided by its contained SwComponentPrototypes
[TPS_SWCT_01202]	ApplicationDataType defines a subset of the values used in the ModeDeclarationGroup
[TPS_SWCT_01203]	PortPrototype may own port annotations
[TPS_SWCT_01204]	GeneralAnnotation
[TPS_SWCT_01205]	Typical annotations for sender/receiver communication
[TPS_SWCT_01206]	Min and Max annotations are valid for a certain amount of time
[TPS_SWCT_01207]	VariableDataPrototypes use the same application-level Sender-ReceiverAnnotation
[TPS_SWCT_01208]	Grouping for SenderReceiverAnnotation
[TPS_SWCT_01209]	ClientServerAnnotation
[TPS_SWCT_01210]	IoHwAbstractionServerAnnotation

[TPS_SWCT_01211]	Assign several annotations to ArgumentDataPrototype
[TPS_SWCT_01212]	ParameterPortAnnotation
[TPS_SWCT_01213]	ModePortAnnotation
[TPS_SWCT_01214]	TriggerPortAnnotation
[TPS_SWCT_01215]	NvDataPortAnnotation
[TPS_SWCT_01216]	DelegatedPortAnnotation
[TPS_SWCT_01217]	Semantics of DelegatedPortAnnotation.signalFan
[TPS_SWCT_01218]	Big picture of ComSpec
[TPS_SWCT_01219]	ComSpec for queued and non-queued sender-receiver communication
[TPS_SWCT_01220]	initValue defines an initial value that shall be taken if the corresponding dataElement has not yet been received
[TPS_SWCT_01221]	DataFilter
[TPS_SWCT_01222]	Applicability of DataFilter
[TPS_SWCT_01223]	networkRepresentation defines how a specific dataElement is represented on a communication bus
[TPS_SWCT_01224]	CompuMethods of dataElement and the networkRepresentation are used for conversion purposes
[TPS_SWCT_01225]	RunnableEntity implements the functionality of two or more ClientServerOperations
[TPS_SWCT_01226]	initValue on the level of a ComSpec is relevant for connections to the corresponding PortPrototype
[TPS_SWCT_01227]	Unconnected RPortPrototype typed by NvDataInterface
[TPS_SWCT_01228]	NvProvideComSpec
[TPS_SWCT_01229]	Three different levels of abstraction regarding the definition of data types
[TPS_SWCT_01230]	Application Data Level
[TPS_SWCT_01231]	Application level may impose strong requirements on the design of the corresponding implementation level
[TPS_SWCT_01232]	Implementation Data Level
[TPS_SWCT_01233]	Use case for the Implementation Data Level
[TPS_SWCT_01234]	Base Level
[TPS_SWCT_01235]	Mapping of data defined on the <i>Application</i> level to the <i>Implementation</i> and <i>Base Type</i> level
[TPS_SWCT_01236]	Big picture of data types
[TPS_SWCT_01237]	SwDataDefProps
[TPS_SWCT_01238]	Attribute category used in the context of AutosarDataType
[TPS_SWCT_01239]	default value for attribute category used in the context of AutosarDataType
[TPS_SWCT_01240]	Subclasses of ApplicationDataType
[TPS_SWCT_01241]	Applicable category s for subclasses ApplicationDataType
[TPS_SWCT_01242]	category characterizes the nature of a data type on application level
[TPS_SWCT_01243]	Definition of enumeration types
[TPS_SWCT_01244]	Data types for calibration parameters are also described as primitive types
[TPS_SWCT_01245]	SwDataDefProps control the structure of calibration parameters
[TPS_SWCT_01246]	SwRecordLayout may be required for A2L generation
[TPS_SWCT_01247]	ApplicationArrayDataType and ApplicationRecordDataType
[TPS_SWCT_01248]	Nested definition of ImplementationDataType
[TPS_SWCT_01249]	ApplicationRecordDataType
[TPS_SWCT_01250]	ImplementationDataType has been introduced to optimize the formal support for data type handling on the implementation level
[TPS_SWCT_01251]	Limited set of values for category are applicable for Implementation-DataType
[TPS_SWCT_01252]	ImplementationDataType can express concepts not available on application level

[TPS_SWCT_01253]	Rules apply for the usage of the attribute ImplementationDataType.type- Emitter
[TPS_SWCT_01254]	ImplementationDataType with array semantics
[TPS_SWCT_01255]	Indicate whether the array is supposed to have a fixed size or whether the actual size might change during run-time
[TPS_SWCT_01256]	Definition of multi-dimensional array data types
[TPS_SWCT_01257]	ImplementationDataType or the aggregated Implementation- DataTypeElements do not form closed sets
[TPS_SWCT_01258]	Definition of a pointer to data
[TPS_SWCT_01259]	Definition of a pointer to a function
[TPS_SWCT_01260]	SwBaseType
[TPS_SWCT_01261]	Use case for SwBaseType
[TPS_SWCT_01262]	memAlignment and byteOrder are platform specific
[TPS_SWCT_01263]	Further use cases for SwBaseType
[TPS_SWCT_01264]	Data prototypes implement a role of a data type
[TPS_SWCT_01265]	DataPrototype aggregates an own set of SwDataDefProps
[TPS_SWCT_01266]	Three non-abstract classes derived from AutosarDataPrototype
[TPS_SWCT_01267]	DataPrototype can be aggregated in different roles
[TPS_SWCT_01268]	Definition of initValue for a VariableDataPrototype or a Parameter- DataPrototype
[TPS_SWCT_01269]	In PortInterfaces , initial values defined for DataPrototypes are ignored
[TPS_SWCT_01270]	AutosarVariableRef
[TPS_SWCT_01271]	AutosarParameterRef
[TPS_SWCT_01272]	Semantics of swComparisonVariable
[TPS_SWCT_01273]	Precedence rules for the application of SwDataDefProps
[TPS_SWCT_01274]	SwDataDefProps used to support calibration and measurement
[TPS_SWCT_01275]	values of the attribute swImplPolicy are restricted depending on the context
[TPS_SWCT_01276]	Computation methods
[TPS_SWCT_01277]	Computation methods are used for the conversion of <i>internal</i> values into their <i>physical</i> representation and vice versa
[TPS_SWCT_01278]	CompuMethods can also be used to assign symbolic names to internal values
[TPS_SWCT_01279]	Preferred conversion direction depends on the use case
[TPS_SWCT_01280]	CompuMethod applied to values outside of its limits
[TPS_SWCT_01281]	Unit associated with a PhysicalDimension
[TPS_SWCT_01283]	Rational function
[TPS_SWCT_01284]	CompuScale might require a representation in the generated RTE C code
[TPS_SWCT_01285]	Physical dimension
[TPS_SWCT_01286]	DataConstr
[TPS_SWCT_01287]	Standard limits and extended limits in the ASAM-MCD2 (ASAP2) specification
[TPS_SWCT_01288]	Interpretation of PhysConstrs and InternalConstrs by tools
[TPS_SWCT_01289]	Semantics of Limit
[TPS_SWCT_01290]	SwAddrMethod
[TPS_SWCT_01291]	Association of MemorySection with SwAddrMethod
[TPS_SWCT_01292]	Usage of SwAddrMethod in the context of a DataPrototype
[TPS_SWCT_01293]	RTE Generator has to derive the Memory Allocation Keyword
[TPS_SWCT_01294]	Missing SwDataDefProps.swAddrMethod
[TPS_SWCT_01295]	SwRecordLayout
[TPS_SWCT_01296]	Different approaches of ASAM MCD-2MC and AUTOSAR with respect to SwRecordLayout
[TPS_SWCT_01297]	Compliance of ApplicationDataTypes or ImplementationDataTypes to swDataDefProps
[TPS_SWCT_01298]	Computing SwRecordLayout from ImplementationDataTypes is not possible

[TPS_SWCT_01299]	Relation of swRecordLayoutGroup to subElement
[TPS_SWCT_01300]	Relationship between record layouts and interpolation routines
[TPS_SWCT_01301]	Importance of initial values
[TPS_SWCT_01302]	Semantics of minimumStartInterval
[TPS_SWCT_01303]	symbol attribute describes the RunnableEntity 's entry point
[TPS_SWCT_01304]	Cat. 1A and 1B RunnableEntity s will eventually terminate
[TPS_SWCT_01305]	RunnableEntity as one that cannot be invoked concurrently
[TPS_SWCT_01306]	Software-component description itself does not put any bounds on the number of concurrent invocations of a RunnableEntity
[TPS_SWCT_01307]	supportsMultipleInstantiation vs. canBeInvokedConcurrently
[TPS_SWCT_01308]	Combination of supportsMultipleInstantiation=false and canBeInvokedConcurrently=false
[TPS_SWCT_01309]	signature of a RunnableEntity depends on the connected RTEEvent
[TPS_SWCT_01310]	Categories of RunnableEntity s
[TPS_SWCT_01311]	Name of an operation argument
[TPS_SWCT_01312]	RunnableEntity has a mapping to BswModuleEntry
[TPS_SWCT_01313]	Conditions for a transition from suspended to to be started
[TPS_SWCT_01314]	RTEEvent
[TPS_SWCT_01315]	Interaction of RunnableEntity with RTEEvent
[TPS_SWCT_01316]	Abstract base class RTEEvent
[TPS_SWCT_01317]	RTE triggers RunnableEntity in response to occurring RTEEvent
[TPS_SWCT_01318]	RunnableEntity and WaitPoint
[TPS_SWCT_01319]	RTEEvent can be used to trigger WaitPoints in different RunnableEntity s
[TPS_SWCT_01320]	RunnableEntity s of category 2
[TPS_SWCT_01321]	Communication among RunnableEntity s
[TPS_SWCT_01322]	Interaction patterns for the application of the sender-receiver paradigm
[TPS_SWCT_01323]	Read and write access to a dataElement
[TPS_SWCT_01324]	Mode switches need to be completed in finite time
[TPS_SWCT_01325]	Read and write access is only applicable for RunnableEntity s of category 1
[TPS_SWCT_01326]	Constrain the scope of a specific communication
[TPS_SWCT_01327]	RTE generator can omit the creation of checks at run-time
[TPS_SWCT_01328]	Default value of attribute scope
[TPS_SWCT_01329]	Access to specific data is implemented by means of aggregating the meta-class VariableAccess in specific roles
[TPS_SWCT_01330]	RunnableEntity can also have dataSendPoints
[TPS_SWCT_01331]	dataWriteAccess vs. dataSendPoint
[TPS_SWCT_01332]	dataReceivePointByValue vs. dataReceivePointByArgument
[TPS_SWCT_01333]	dataReceivePointByValue/dataReceivePointByArgument vs. dataReadAccess
[TPS_SWCT_01334]	RunnableEntity s of category 1 may have dataReceivePointByValues/dataReceivePointByArguments
[TPS_SWCT_01335]	Combine dataReceivePointByValue or dataReceivePointByArgument with a WaitPoint
[TPS_SWCT_01336]	dataSendPoint also allows for the definition of a DataSendCompletedEvent
[TPS_SWCT_01337]	DataReceivedEvent
[TPS_SWCT_01338]	DataReceiveErrorEvent
[TPS_SWCT_01339]	RTE activates RunnableEntity in response to DataReceiveErrorEvent
[TPS_SWCT_01340]	DataReceiveErrorEvent cannot be combined with a WaitPoint
[TPS_SWCT_01341]	DataReceiveErrorEvent is directly associated with the corresponding VariableDataPrototype
[TPS_SWCT_01342]	Invocation of a server operation

[TPS_SWCT_01343]	Synchronous vs. asynchronous invocation
[TPS_SWCT_01344]	Consistency of values of timeout
[TPS_SWCT_01345]	Synchronous operation invocation
[TPS_SWCT_01346]	Asynchronous operation invocation
[TPS_SWCT_01347]	Blocking access to operation result in an asynchronous operation invocation
[TPS_SWCT_01348]	Trigger source
[TPS_SWCT_01349]	Trigger sink
[TPS_SWCT_01350]	Calibration Parameters shared among several SwComponentTypes
[TPS_SWCT_01351]	Access to a ParameterDataPrototype
[TPS_SWCT_01352]	Requested mode is just sent and received as an ordinary data value
[TPS_SWCT_01353]	RunnableEntity s react on a mode request via a corresponding RTEEvent
[TPS_SWCT_01354]	PortAPIOption
[TPS_SWCT_01355]	enableTakeAddress = true
[TPS_SWCT_01356]	indirectAPI option switches the generation of the RTE's indirect API functionality
[TPS_SWCT_01357]	Definition of implicit values that are passed by the RTE to the server's entry point
[TPS_SWCT_01358]	Values are hidden from the client components
[TPS_SWCT_01359]	Private memory per instance
[TPS_SWCT_01360]	Arbitrary number of per-instance memory blocks
[TPS_SWCT_01361]	attribute supportsMultipleInstantiation == false
[TPS_SWCT_01362]	Initialization of PerInstanceMemory
[TPS_SWCT_01363]	PerInstanceMemory typed by 'C' Data Types
[TPS_SWCT_01364]	Initial value of a PerInstanceMemory typed by 'C' Data Types
[TPS_SWCT_01365]	PerInstanceMemory typed by AUTOSAR Data Types
[TPS_SWCT_01366]	Initial value of a PerInstanceMemory typed by AUTOSAR Data Types
[TPS_SWCT_01367]	Typed by AUTOSAR data type vs. typed by C data type
[TPS_SWCT_01368]	Describe static and constant memory
[TPS_SWCT_01369]	Static and constant memory is not instantiated by the RTE
[TPS_SWCT_01370]	VariationPointProxy
[TPS_SWCT_01371]	VariationPointProxy vs. VariationPoint
[TPS_SWCT_01372]	bindingTime = PreCompileTime
[TPS_SWCT_01373]	RTE generator shall evaluate the SwSystemconstDependentFormula
[TPS_SWCT_01374]	Implementation of AutosarParameterRef
[TPS_SWCT_01375]	Implementation of AutosarVariableRef
[TPS_SWCT_01376]	Software-components need to be capable of reacting to state changes
[TPS_SWCT_01377]	Two mechanisms to define how SwcInternalBehavior should interact with the mode management
[TPS_SWCT_01378]	AtomicSwComponentType can define an SwcModeSwitchEvent to execute RunnableEntity
[TPS_SWCT_01379]	AtomicSwComponentType can indicate whether an RTEEvent that starts an associated RunnableEntity is disabled in a certain mode
[TPS_SWCT_01380]	Mode management behavior on the sender side
[TPS_SWCT_01381]	Read the currently active mode
[TPS_SWCT_01382]	Mode switch requests are handled asynchronously by the RTE
[TPS_SWCT_01383]	ModeSwitchPoint
[TPS_SWCT_01384]	Execution of initialization code for software-components
[TPS_SWCT_01385]	Execution of initialization code for software-components
[TPS_SWCT_01386]	Initialization by mode management
[TPS_SWCT_01387]	Finalization by mode management
[TPS_SWCT_01388]	Initial modes of AtomicSwComponentTypes are defined by the initialMode
[TPS_SWCT_01389]	I/O Hardware Abstraction interfaces MCAL drivers

[TPS_SWCT_01390]	I/O Hardware Abstraction might have sub-structures
[TPS_SWCT_01391]	I/O Hardware Abstraction abstracts from the location of peripheral I/O devices
[TPS_SWCT_01392]	Mapping between the EcuAbstractionSwComponentType and the corresponding BswModuleDescription
[TPS_SWCT_01393]	Complex Driver
[TPS_SWCT_01394]	Complex Driver is represented by the ComplexDeviceDriverSwComponentType
[TPS_SWCT_01395]	ComplexDeviceDriverSwComponentType has dependencies to ECU Hardware
[TPS_SWCT_01396]	Mapping between the ComplexDeviceDriverSwComponentType and the corresponding BswModuleDescription
[TPS_SWCT_01397]	Hybrid concept between Basic Software Modules and a SwComponentType
[TPS_SWCT_01398]	Communication patterns for AUTOSAR services
[TPS_SWCT_01399]	Dependency is modeled by aggregating required and provided PortPrototypes
[TPS_SWCT_01400]	PortInterface selected from the set of standardized Service Interfaces
[TPS_SWCT_01401]	Form a top-level RootSwCompositionPrototype
[TPS_SWCT_01402]	Mapping of all AtomicSwComponentType instances to EcuInstances
[TPS_SWCT_01403]	Impact of AUTOSAR services on the methodology
[TPS_SWCT_01404]	Creation of the EcuExtract
[TPS_SWCT_01405]	Creation of the ServiceSwComponentTypes
[TPS_SWCT_01406]	Creation of SwComponentPrototype typed by a ServiceSwComponentType
[TPS_SWCT_01407]	Creation of InternalBehavior typed by a ServiceSwComponentType
[TPS_SWCT_01408]	Creation of SwcBswMapping
[TPS_SWCT_01409]	Update of PortDefinedArgumentValues
[TPS_SWCT_01410]	Dcm and Dem can directly access dataElements in PPortPrototypes typed by a SenderReceiverInterface
[TPS_SWCT_01411]	Use cases for a ServiceSwComponentType to express ServiceNeeds
[TPS_SWCT_01412]	ServiceSwComponentType shall be added in ECU Configuration phase
[TPS_SWCT_01413]	Local communication with services
[TPS_SWCT_01414]	Mode manager needs to communicate with application software components located on other ECUs
[TPS_SWCT_01415]	Interfaces of ServiceProxySwComponentType
[TPS_SWCT_01416]	Difference between a ServiceProxySwComponentType and an ApplicationSwComponentType
[TPS_SWCT_01417]	Define calibration parameters common to all SwComponentPrototypes of the same SwComponentType
[TPS_SWCT_01418]	Ways to define a calibration parameter
[TPS_SWCT_01419]	ParameterSwComponentType shall never aggregate a SwcInternalBehavior
[TPS_SWCT_01420]	SwComponentType requiring access to shared calibration parameters needs RPortPrototype typed by a ParameterInterface
[TPS_SWCT_01421]	ParameterInterface is not restricted to parameters which can actually can be calibrated
[TPS_SWCT_01422]	Delegation of PortPrototypes typed by a ParameterInterface
[TPS_SWCT_01423]	ParameterDataPrototype aggregated in the role constantMemory
[TPS_SWCT_01424]	ParameterDataPrototype aggregated in the role perInstanceParameter

[TPS_SWCT_01425]	AtomicSwComponentType provides one callback per event if diagnostic event data change
[TPS_SWCT_01426]	AtomicSwComponentType provides callback if any diagnostic event data and/or status changed
[TPS_SWCT_01427]	AtomicSwComponentType provides data for diagnostic purposes via ClientServerInterface
[TPS_SWCT_01428]	ServiceSwComponentType representing the Dem provides a PPortPrototype for the Dcm
[TPS_SWCT_01429]	[constr_1135] only applies for BITFIELD_TEXTTABLE
[TPS_SWCT_01430]	Conversion specification from internal to physical values as well as the reverse conversion
[TPS_SWCT_01431]	Finding the symbol for the representation of a CompuScale in C code
[TPS_SWCT_01432]	Keep the invalidValue transparent to the sending and receiving software components
[TPS_SWCT_01433]	Invalid values outside the range limits
[TPS_SWCT_01434]	Sender and receiver have knowledge of invalid value
[TPS_SWCT_01435]	Invalid values outside the range limits
[TPS_SWCT_01436]	Different receivers require different handling of data invalidation
[TPS_SWCT_01437]	invalidValue can also be specified without setting a compuMethod
[TPS_SWCT_01438]	Handling of invalidation in the sending RTE
[TPS_SWCT_01439]	Handling of invalidation in the receiving RTE
[TPS_SWCT_01440]	Measurement is not limited to primitive objects
[TPS_SWCT_01441]	Nature of a TYPE_REFERENCE
[TPS_SWCT_01442]	ImplementationDataType of category TYPE_REFERENCE does not define own properties
[TPS_SWCT_01443]	ImplementationDataType of category TYPE_REFERENCE overwrites properties of refined ImplementationDataType
[TPS_SWCT_01444]	Size of SwBaseType is specified in bits
[TPS_SWCT_01445]	Applicability of SwDataDefProps for DataPrototypes
[TPS_SWCT_01446]	References to a DataPrototype may or may not imply the necessity for using an instanceRef
[TPS_SWCT_01447]	Applicable binding times for model elements in the scope of the Software Component Template
[TPS_SWCT_01448]	Pre-defined values for the category of VariationPointProxy
[TPS_SWCT_02000]	Default value for attribute swImplPolicy
[TPS_SWCT_02001]	Values of SwAxisCont with the category CURVE_AXIS , COM_AXIS , RES_AXIS are for display only
[TPS_SWCT_02002]	AtomicSwComponentType offers a PPortPrototypes typed by ClientServerInterface to read/write current value via diagnostic services
[TPS_SWCT_02003]	AtomicSwComponentType offers PortPrototypes typed by Sender-ReceiverInterfaces to read/write current values via diagnostic services
[TPS_SWCT_02004]	AtomicSwComponentType offers a PortPrototype typed by a ClientServerInterface to start/stop or request routine results of diagnostic routines
[TPS_SWCT_02005]	ARMetaClassAtomicSwComponentType offers PortPrototypes typed by ClientServerInterfaces to adjust the IO signal via diagnostic services
[TPS_SWCT_02006]	AtomicSwComponentType offers sender receiver ports to adjust the IO signal via diagnostic services
[TPS_SWCT_02007]	AtomicSwComponentType implements a OBD system monitor with In-Use-Monitor Performance Ratio
[TPS_SWCT_02008]	AtomicSwComponentType offers a server port to read/write current value via OBD services

[TPS_SWCT_02009]	AtomicSwComponentType offers sender receiver ports to read/write current values via OBD services
[TPS_SWCT_02010]	AtomicSwComponentType offers a server port to read vehicle information values via OBD services
[TPS_SWCT_02011]	AtomicSwComponentType offers a server port to read DTR value via OBD services
[TPS_SWCT_02012]	AtomicSwComponentType offers a server port for request control of on-board system, test or component via OBD services
[TPS_SWCT_02013]	AtomicSwComponentType offers a server port to get protocol, session and security information or to request a Reset to Default Session
[TPS_SWCT_02014]	AtomicSwComponentType supports Response On Event (ROE) via diagnostic services
[TPS_SWCT_02015]	AtomicSwComponentType verifies the access to security level via diagnostic services
[TPS_SWCT_02016]	AtomicSwComponentType requires information on the status of the protocol communication and may disallow a protocol
[TPS_SWCT_02017]	AtomicSwComponentType requires the notification about a Service Request via diagnostic services
[TPS_SWCT_02018]	Setup for AtomicSwComponentType which contains a Supervised Entity
[TPS_SWCT_02019]	Setup for AtomicSwComponentType which requires <i>Global Supervision Status</i> notification
[TPS_SWCT_02020]	AtomicSwComponentType uses the hash calculation of the Crypto Service
[TPS_SWCT_02021]	AtomicSwComponentType uses the message authentication code (MAC) calculation of the Crypto Service
[TPS_SWCT_02022]	AtomicSwComponentType uses the message authentication code (MAC) verification of the Crypto Service
[TPS_SWCT_02023]	AtomicSwComponentType uses the generation of random seed of the Crypto Service
[TPS_SWCT_02024]	AtomicSwComponentType uses the generation of random numbers of the Crypto Service
[TPS_SWCT_02025]	AtomicSwComponentType uses the symmetrical block encryption of the Crypto Service
[TPS_SWCT_02026]	AtomicSwComponentType uses the symmetrical block decryption of the Crypto Service
[TPS_SWCT_02027]	AtomicSwComponentType uses the symmetrical encryption of the Crypto Service
[TPS_SWCT_02028]	AtomicSwComponentType uses the symmetrical decryption of the Crypto Service
[TPS_SWCT_02029]	AtomicSwComponentType uses the asymmetrical encryption of the Crypto Service
[TPS_SWCT_02030]	AtomicSwComponentType uses the asymmetrical decryption of the Crypto Service
[TPS_SWCT_02031]	AtomicSwComponentType uses the signature generation of the Crypto Service
[TPS_SWCT_02032]	AtomicSwComponentType uses the signature verification of the Crypto Service
[TPS_SWCT_02033]	AtomicSwComponentType uses the checksum calculation of the Crypto Service
[TPS_SWCT_02034]	AtomicSwComponentType uses the key derivation of the Crypto Service
[TPS_SWCT_02035]	AtomicSwComponentType uses the symmetric key derivation of the Crypto Service
[TPS_SWCT_02036]	AtomicSwComponentType uses the key exchange interface for public value calculation of the Crypto Service

[TPS_SWCT_02037]	AtomicSwComponentType uses the key exchange interface for secret value calculation of the Crypto Service
[TPS_SWCT_02038]	AtomicSwComponentType uses the key exchange interface to calculate symmetric key with the Crypto Service
[TPS_SWCT_02039]	AtomicSwComponentType uses the symmetrical key extraction of the Crypto Service
[TPS_SWCT_02040]	AtomicSwComponentType uses the symmetrical key wrapping of the Crypto Service to export a symmetrical key structure with a symmetric key
[TPS_SWCT_02041]	AtomicSwComponentType uses the asymmetrical key wrapping of the Crypto Service to export a symmetrical key structure with a asymmetric key
[TPS_SWCT_02042]	AtomicSwComponentType uses the asymmetrical public key extraction of the Crypto Service
[TPS_SWCT_02043]	AtomicSwComponentType uses the asymmetrical private key extraction of the Crypto Service
[TPS_SWCT_02044]	AtomicSwComponentType uses the asymmetrical key wrapping of the Crypto Service to export a (asymmetric) private key structure with a symmetrical wrapping key
[TPS_SWCT_02045]	AtomicSwComponentType uses the asymmetrical key wrapping of the Crypto Service to export a (asymmetric) private key structure with a asymmetrical wrapping key

Table C.7: Added Specification Items in 4.0.3

C.3.4 Deleted Constraints in R4.0.3

Number	Heading
[constr_1023]	Specification of Units in CompuMethods ¹
[constr_1062]	Compatibility of data types with category BIT
[constr_1122]	Existence of attributes in PROFILE_03
[constr_1123]	Constraints of dataLength in PROFILE_03
[constr_1124]	Constraints of dataId in PROFILE_03
[constr_1125]	Constraints of maxDeltaCounterInit in PROFILE_03
[constr_1127]	ServiceSwComponentType shall not have ServiceNeeds
[constr_1136]	Compatibility of introduction of blueprint and blueprinted element
	the following constraints are moved to [1]
[constr_2500]	PortInterfaces shall be of same kind
[constr_2526]	PortInterfaces need to be compatible to the blueprints
[constr_2527]	Blueprints shall live in package of a proper category
[constr_2528]	PortPrototypes shall not refer to blueprints of PortInterfaces
[constr_2529]	Blueprints of ports and interfaces shall be compatible
[constr_4001]	Content of ModeRequestTypeMap

Table C.8: Deleted Constraints in R4.0.3

C.3.5 Deleted Specification Items

N/A

¹The text is still there but it does no longer represent a constraint.

C.4 Constraint History of this Document according to AUTOSAR R4.1.1

C.4.1 Changed Constraints in R4.1.1

Please note that constraints listed using an bold typeface have been deleted in later versions of the document.

Number	Heading
[constr_1012]	Value of <code>category</code> is <code>FIXED_LENGTH</code>
[constr_1013]	Value of <code>category</code> is <code>VARIABLE_LENGTH</code>
[constr_1016]	Restriction of <code>invalidValue</code> for <code>ImplementationDataType</code> and <code>ImplementationDataTypeElement</code>
[constr_1026]	Compatibility of <code>Units</code>
[constr_1047]	Compatibility of <code>ApplicationPrimitiveDataTypes</code>
[constr_1048]	Compatibility of <code>ApplicationRecordDataTypes</code>
[constr_1049]	Compatibility of <code>ApplicationArrayDataTypes</code>
[constr_1050]	Compatibility of <code>ImplementationDataTypes</code>
[constr_1060]	Compatibility of data types with <code>category</code> <code>ARRAY</code> , <code>VAL_BLK</code>
[constr_1072]	Compatibility of <code>ModeSwitchInterfaces</code> in the context of an <code>AssemblySwConnector</code>
[constr_1073]	Compatibility of <code>ModeSwitchInterfaces</code> in the context of an <code>DelegationSwConnector</code>
[constr_1074]	Compatibility of <code>ModeDeclarationGroupPrototypes</code>
[constr_1075]	Compatibility of <code>ModeDeclarationGroups</code>
[constr_1079]	Compatibility of <code>ClientServerInterfaces</code> in the context of an <code>AssemblySwConnector</code>
[constr_1080]	Compatibility of <code>ClientServerInterfaces</code> in the context of an <code>DelegationSwConnector</code>
[constr_1081]	Compatibility of <code>TriggerInterfaces</code> in the context of an <code>AssemblySwConnector</code>
[constr_1082]	Compatibility of <code>TriggerInterfaces</code> in the context of an <code>DelegationSwConnector</code>
[constr_1068]	Compatibility of <code>VariableDataPrototypes</code> or <code>ParameterDataPrototypes</code> typed by primitive data types
[constr_1069]	Compatibility of <code>PortPrototypes</code> of different <code>DataInterfaces</code> in the context of <code>AssemblySwConnectors</code>
[constr_1070]	Compatibility of <code>PortPrototypes</code> of different <code>DataInterfaces</code> in the context of <code>DelegationSwConnectors</code>
[constr_1072]	Compatibility of <code>ModeSwitchInterfaces</code> in the context of an <code>AssemblySwConnector</code>
[constr_1073]	Compatibility of <code>ModeSwitchInterfaces</code> in the context of an <code>DelegationSwConnector</code>
[constr_1074]	Compatibility of <code>ModeDeclarationGroupPrototypes</code>
[constr_1079]	Compatibility of <code>ClientServerInterfaces</code> in the context of an <code>AssemblySwConnector</code>
[constr_1080]	Compatibility of <code>ClientServerInterfaces</code> in the context of an <code>DelegationSwConnector</code>
[constr_1081]	Compatibility of <code>TriggerInterfaces</code> in the context of an <code>AssemblySwConnector</code>
[constr_1082]	Compatibility of <code>TriggerInterfaces</code> in the context of an <code>DelegationSwConnector</code>
[constr_1108]	Value of <code>ApplicationError.errorCode</code>
[constr_1177]	Allowed <code>targetCategory</code> for <code>SwPointerTargetProps</code>

[constr_1187]	Compatibility of VariableDataPrototypes or ParameterDataPrototypes typed by composite data types

Table C.9: Changed Constraints in R4.1.1

C.4.2 Added Constraints in R4.1.1

Number	Heading
[constr_1191]	Value of Limit shall yield a numerical value
[constr_1192]	Compatibility of “ IDENTICAL ” to “ RAT_FUNC ” or “ LINEAR ”
[constr_1193]	ModeDeclaration shall be referenced by at least one ModeTransition in the role enteredMode
[constr_1194]	Identical ModeTransitions
[constr_1195]	SwcModeSwitchEvent and the definition of ModeTransition
[constr_1196]	Existence of networkRepresentation vs. compositeNetworkRepresentation
[constr_1197]	Existence of compositeNetworkRepresentation shall be comprehensive
[constr_1200]	Queued communication is not applicable for dataElements owned by PRPortPrototype
[constr_1201]	initValue shall exist in an RPortPrototype
[constr_1202]	Supported connections by AssemblySwConnector for PortPrototypes typed by a SenderReceiverInterface or NvDataInterface
[constr_1203]	Supported connections by DelegationSwConnector for PortPrototypes typed by a SenderReceiverInterface or NvDataInterface
[constr_1204]	Supported connections by AssemblySwConnector for PortPrototypes typed by a ClientServerInterface , ModeSwitchInterface , or TriggerInterface
[constr_1205]	Supported connections by DelegationSwConnector for PortPrototypes typed by a ClientServerInterface , ModeSwitchInterface , or TriggerInterface
[constr_1209]	Mapping of ModeDeclarations of mode user to ModeDeclaration of mode manager
[constr_1210]	Mapping of ModeDeclarations of mode user to all ModeDeclarations of mode manager
[constr_1211]	Constraints of maxNoNewOrRepeatedData in PROFILE_01
[constr_1212]	Constraints of syncCounterInit in PROFILE_01
[constr_1213]	Constraints of maxNoNewOrRepeatedData in PROFILE_02
[constr_1214]	Constraints of syncCounterInit in PROFILE_02
[constr_1215]	Interpretation of attribute maxNoNewOrRepeatedData owned by EndToEndDescription in PROFILE_01
[constr_1216]	Interpretation of attribute syncCounterInit owned by EndToEndDescription in PROFILE_01
[constr_1217]	Interpretation of attribute maxNoNewOrRepeatedData owned by EndToEndDescription in PROFILE_02
[constr_1218]	Interpretation of attribute syncCounterInit owned by EndToEndDescription in PROFILE_02
[constr_1219]	Invalidation depends on the value of swImplPolicy
[constr_1220]	Compatibility of SwBaseType
[constr_1221]	DataPrototype is typed by an ApplicationPrimitiveDataType
[constr_1222]	category of an AutosarDataType used to type a DataPrototype is set to STRING
[constr_1223]	DataPrototype is typed by an ApplicationRecordDataType

[constr_1224]	DataPrototype is typed by an ApplicationArrayDataType
[constr_1225]	DataPrototype is typed by an ImplementationDataType that references a CompuMethod of category TEXTTABLE or BITFIELD_TEXTTABLE
[constr_1226]	Applicable range for ExecutableEntityActivationReason.bitPosition
[constr_1227]	Value of attribute ExecutableEntityActivationReason.bitPosition shall be unique
[constr_1228]	RTEEvent that is referenced by a WaitPoint in the role trigger shall not reference ExecutableEntityActivationReason
[constr_1229]	category of ImplementationDataType boils down to VALUE
[constr_1230]	ApplicationDataType that qualifies for Integral Primitive Type
[constr_1231]	ConsistencyNeeds aggregated by CompositionSwComponentType
[constr_1232]	ConsistencyNeeds aggregated by AtomicSwComponentType
[constr_1233]	InstantiationTimingEventProps shall only reference TimingEvent
[constr_1234]	Value of RunnableEntity.symbol
[constr_1237]	Scope of mapped ClientServerOperations in the context of a ClientServer-OperationMapping
[constr_1238]	Scope of mapped ApplicationErrors in the context of a ClientServerOperationMapping
[constr_1239]	RuleBasedValueSpecification shall not exceed the number of values required
[constr_1240]	Consistency of ArgumentDataPrototypes within the context of a ClientServer-OperationMapping
[constr_1241]	Compound Primitive Data Types and invalidValue
[constr_1242]	Restriction of invalidValue for ApplicationPrimitiveDataType
[constr_1243]	NumericalOrText shall either define vf or vt
[constr_1244]	DataPrototypes used in application software shall not be typed by C enums
[constr_1245]	Consideration of ModeTransitions for the compatibility of ModeDeclarationGroups
[constr_1246]	Consistency of firstMode and secondMode in the scope of one ModeDeclarationMappingSet
[constr_1247]	Consistency of ModeDeclarationMappingSet with respect to the referenced firstModeGroup and secondModeGroup
[constr_1248]	Compatibility of PortPrototypes of different DataInterfaces in the context of a PassThroughSwConnector
[constr_1249]	Compatibility of ModeSwitchInterfaces in the context of a PassThroughSwConnector
[constr_1250]	Compatibility of ClientServerInterfaces in the context of a PassThroughSwConnector
[constr_1251]	Compatibility of PortPrototypes of TriggerInterfaces in the context of a PassThroughSwConnector
[constr_1252]	Creation of a loop involving a PassThroughSwConnector is not allowed
[constr_1253]	Supported usage of VariationPointProxy
[constr_1254]	Definition of a pointer to a pointer
[constr_1255]	ApplicationPrimitiveDataTypes of category BOOLEAN and STRING
[constr_1256]	Acknowledgement feedback in n:1 writer case
[constr_1257]	No WaitPoints allowed
[constr_1258]	Value of minimumStartInterval for RunnableEntities triggered by an InitEvent
[constr_1259]	Aggregation of AsynchronousServerCallPoint and AsynchronousServer-CallResultPoint
[constr_1260]	No mode disabling for InitEvents
[constr_1261]	Applicability for EndToEndDescription.dataIdNibbleOffset
[constr_1263]	Existence of ModeErrorBehavior.defaultMode
[constr_1264]	Iteration along output axis is only supported for VALUE and VAL_BLK

[constr_1268]	ArgumentDataPrototype.direction shall be preserved in a ClientServerOperationMapping
[constr_1269]	Number of arguments shall be preserved in a ClientServerOperationMapping
[constr_1270]	ArgumentDataPrototype shall be mapped only once in a ClientServerOperationMapping
[constr_1271]	ArrayValueSpecification.elements shall be identical to the number of ApplicationRecordDataType.element
[constr_1272]	ArrayValueSpecification.elements shall be identical to the number of subElements of ImplementationDataType of category STRUCTURE
[constr_1273]	ArrayValueSpecification.elements shall be identical to the value of ApplicationArrayDataType.element.maxNumberOfElements
[constr_1274]	ArrayValueSpecification.elements shall be identical to the value of ImplementationDataType.subElement.arraySize of category ARRAY
[constr_2054]	Valid targets of rptSystem
[constr_2055]	Valid targets of byPassPoint and rptHook reference
[constr_2056]	Consistency of RapidPrototypingScenario with respect to rptSystem and rptArHook references
[constr_2057]	Mandatory information of a RuleBasedAxisCont
[constr_2058]	Mandatory information of a RuleBasedValueCont
[constr_4082]	RunnableEntity.reentrancyLevel shall not be set.

Table C.10: Added Constraints in R4.1.1

Please note that [constr_2533] has been retagged to [constr_1264] to fix a duplicate constraint ID.

C.4.3 Changed Specification Items in R4.1.1

Number	Heading
[TPS_SWCT_01000]	Usage of attribute symbol of the symbolProps
[TPS_SWCT_01001]	Prefix symbols generated for the RunnableEntity
[TPS_SWCT_01085]	Variation on the behavior level
[TPS_SWCT_01112]	Semantics of PortPrototypes
[TPS_SWCT_01113]	Connecting two PortPrototypes
[TPS_SWCT_01128]	SwRecordLayout needed for ApplicationPrimitiveDataType of category STRING
[TPS_SWCT_01179]	Compound Primitive Data Type
[TPS_SWCT_01368]	Describe static and constant memory

Table C.11: Changed Specification Items in R4.1.1

C.4.4 Added Specification Items in R4.1.1

Number	Heading
[TPS_SWCT_01448]	Pre-defined values for the category of VariationPointProxy
[TPS_SWCT_01449]	Semantics of a ModeDeclarationGroupPrototypeMapping
[TPS_SWCT_01450]	Semantics of a ModeTransition
[TPS_SWCT_01451]	Relations between ModeTransition and ModeDeclaration
[TPS_SWCT_01452]	Applicability of networkRepresentation for ApplicationCompositeDataType

[TPS_SWCT_01454]	PRPortPrototype can own both RPortComSpecs and PPortComSpecs
[TPS_SWCT_01455]	Duplicate existence of <code>initValue</code> in the context of a PRPortPrototype
[TPS_SWCT_01456]	Predefined values for <code>MemorySection.option</code> and <code>SwAddrMethod.option</code>
[TPS_SWCT_01457]	<code>ExclusiveAreaNestingOrder</code>
[TPS_SWCT_01458]	Indicate that the locking behavior is fully described for <code>RunnableEntity</code>
[TPS_SWCT_01459]	Locking behavior is not described for this <code>RunnableEntity</code>
[TPS_SWCT_01460]	Relation of <code>SynchronousServerCallPoint</code> to <code>ExclusiveAreaNestingOrder</code>
[TPS_SWCT_01461]	Existence of <code>ImplementationDataType</code>
[TPS_SWCT_01462]	<code>ModeDeclarationMapping</code> defines the explicit correlation of <code>ModeDeclarations</code>
[TPS_SWCT_01463]	<code>modeDeclarationMapping</code> defines the applicable set of <code>ModeDeclarationMappings</code>
[TPS_SWCT_01464]	<code>ModeDeclaration</code> of a mode user is mapped to exactly one <code>ModeDeclaration</code> of a mode manager
[TPS_SWCT_01465]	<code>ModeDeclaration</code> of a mode user is mapped to several <code>ModeDeclarations</code> of a mode manager
[TPS_SWCT_01466]	<code>ConsistencyNeeds</code> applied on <code>RunnableEntities</code> that do not use implicit communication
[TPS_SWCT_01467]	<code>ImplementationDataType</code> references an <code>SwBaseType</code> with a string encoding
[TPS_SWCT_01469]	RTE API for retrieving the current activation reason
[TPS_SWCT_01470]	<code>RunnableEntityGroup</code>
[TPS_SWCT_01471]	<code>DataPrototypeGroup</code>
[TPS_SWCT_01472]	Receiving <code>SwComponentType</code> owns a <code>DataPrototypeGroup</code> in the role <code>dpgRequiresStability</code>
[TPS_SWCT_01473]	Receiving <code>SwComponentType</code> owns a <code>RunnableEntityGroup</code> in the role <code>regRequiresStability</code>
[TPS_SWCT_01474]	Receiving <code>SwComponentType</code> owns a <code>RunnableEntityGroup</code> in the role <code>regRequiresStability</code> and also owns one or several <code>DataPrototypeGroups</code> in the role <code>dpgRequiresStability</code>
[TPS_SWCT_01475]	Sending <code>SwComponentType</code> owns a <code>DataPrototypeGroup</code> in the role <code>dpgRequiresStability</code>
[TPS_SWCT_01476]	Sender and receiver of the same implicitly communicated <code>VariableDataPrototypes</code> are associated with the same <code>RunnableEntityGroup</code>
[TPS_SWCT_01477]	Integral Primitive Types
[TPS_SWCT_01478]	Array size is defined as an attribute of the <code>ImplementationDataTypeElement</code>
[TPS_SWCT_01479]	Applicability of <code>ConsistencyNeeds</code>
[TPS_SWCT_01480]	<i>Stability</i> and/or <i>coherence</i> is not required
[TPS_SWCT_01481]	The meaning of the term <i>stability</i> with respect to <code>ConsistencyNeeds</code>
[TPS_SWCT_01482]	The meaning of the term <i>coherence</i> with respect to <code>ConsistencyNeeds</code>
[TPS_SWCT_01483]	Use static and constant memory to support Measurement and Calibration
[TPS_SWCT_01484]	Meaning of <code>ApplicationRuleBasedValueSpecification</code>
[TPS_SWCT_01485]	<code>RuleBasedValueSpecification</code> shall initialize first elements in an array
[TPS_SWCT_01486]	<code>ApplicationPrimitiveDataType</code> of category <code>STRING</code> may have <code>invalidValue</code>
[TPS_SWCT_01487]	Correspondence of <code>invalidValue</code> for <code>ApplicationPrimitiveDataType</code> and <code>ImplementationDataType</code>
[TPS_SWCT_01488]	<code>ApplicationPrimitiveDataType</code> shall be interpreted as a string of a particular encoding
[TPS_SWCT_01489]	Standardized values of <code>SwRecordLayoutV.swRecordLayoutVProp</code>

[TPS_SWCT_01490]	AUTOSAR supports ApplicationErrors only for ClientServerInterfaces
[TPS_SWCT_01491]	AUTOSAR system does not need to explicitly describe infrastructure errors
[TPS_SWCT_01492]	Default values for factorSiToUnit and offsetSiToUnit
[TPS_SWCT_01493]	The number of RuleArguments.arguments shall not exceed the array size
[TPS_SWCT_01494]	A RuleBasedValueSpecification of rule <code>FILL_UNITIL_END</code> shall fill the value of the last RuleArguments.argument until the last element of the array
[TPS_SWCT_01495]	Standardized value of RuleBasedValueSpecification.category
[TPS_SWCT_01496]	General precedence rule for attributes of SwDataDefProps
[TPS_SWCT_01497]	Precedence of the unit of value axis
[TPS_SWCT_01498]	Precedence of the DataConstr of value axis
[TPS_SWCT_01499]	Precedence of the CompuMethod of value axis
[TPS_SWCT_01500]	Precedence of the display format of value axis
[TPS_SWCT_01501]	Precedence of the calibration access of value axis
[TPS_SWCT_01502]	Precedence of the Unit of the input axis
[TPS_SWCT_01503]	Precedence of the DataConstr of the input axis
[TPS_SWCT_01504]	Precedence of the display format of the input axis
[TPS_SWCT_01505]	Precedence of calibration access along structure hierarchies in complex types
[TPS_SWCT_01506]	Precedence of the calibration access of input axis
[TPS_SWCT_01507]	The role of PassThroughSwConnector
[TPS_SWCT_01508]	Scope of end-to-end protection
[TPS_SWCT_01509]	Implicit communication behavior
[TPS_SWCT_01510]	The role of pretended networking
[TPS_SWCT_01511]	Configuration option is encoded into ModeDeclaration
[TPS_SWCT_01512]	Request change of Pretended Networking mode
[TPS_SWCT_01513]	React on the change of Pretended Networking mode
[TPS_SWCT_01514]	Duplicate existence of initValue in the context of a PRPortPrototype
[TPS_SWCT_01515]	PPortInCompositionInstanceRef shall be used for attaching DelegationSwConnector to an inner PRPortPrototype
[TPS_SWCT_01516]	PortInterface describes the static structure of information interchange
[TPS_SWCT_01517]	ClientServerOperation cannot be passed as a reference
[TPS_SWCT_01518]	Priority of initial value definition with respect to conceptual levels
[TPS_SWCT_01519]	RTE executes certain RunnableEntity periodically
[TPS_SWCT_01520]	Implication of the existence of possibleError on compatibility of ClientServerOperations
[TPS_SWCT_01521]	Use AutosarVariableRef.localVariable for referencing inter-runnable variables
[TPS_SWCT_01522]	No initial value is specified for implicitInterRunnableVariable or explicitInterRunnableVariable
[TPS_SWCT_01523]	Internal trigger event
[TPS_SWCT_01524]	Usage of IoHwAbstractionServerAnnotation
[TPS_SWCT_01525]	InitEvent references a RunnableEntity in the role startOnEvent
[TPS_SWCT_01528]	Meaning of NumericalRuleBasedValueSpecification
[TPS_SWCT_01529]	Default value for EndToEndDescription.dataIdNibbleOffset
[TPS_SWCT_01530]	Error behavior of mode manager and mode user
[TPS_SWCT_01531]	The semantics of ModeErrorReactionPolicyEnum
[TPS_SWCT_01532]	The role of ModeErrorBehavior.defaultMode
[TPS_SWCT_01533]	ModeDeclarationGroup.initialMode shall be assumed in the absence of ModeDeclarationGroup.modeManagerErrorBehavior
[TPS_SWCT_01534]	ModeDeclarationGroup.initialMode shall be assumed in the absence of ModeDeclarationGroup.modeUserErrorBehavior
[TPS_SWCT_01535]	Mode manager reacts on mode error

[TPS_SWCT_01536]	Coherent behavior of all mode users in case of errors in the mode switch communication
[TPS_SWCT_01541]	Preferential selection of modeUserErrorBehavior
[TPS_SWCT_01542]	Preferential selection of modeManagerErrorBehavior
[TPS_SWCT_01543]	PortInterfaceMapping overrides all other compatibility rules
[TPS_SWCT_01544]	<i>prefix</i> used for the actual name of the used PortInterface for the routing activation
[TPS_SWCT_01545]	ModeDeclaration of a <i>mode user</i> that is not mapped to a ModeDeclaration of a <i>mode manager</i>
[TPS_SWCT_01546]	Notification when an external tester is attached or activated
[TPS_SWCT_01547]	Ability to set and reset the Warning Indicator Status bit
[TPS_SWCT_02046]	byPassPoint specifies the rapid prototyping capability
[TPS_SWCT_02047]	rptHook specifies the link to rapid prototyping algorithm
[TPS_SWCT_02048]	Implicit SwComponentPrototype selection for Rapid Prototyping Scenario
[TPS_SWCT_02049]	Implicit RunnableEntity selection for Rapid Prototyping Scenario
[TPS_SWCT_02050]	Explicit access point selection for Rapid Prototyping Scenario
[TPS_SWCT_02051]	Explicit DataPrototype selection for Rapid Prototyping Scenario
[TPS_SWCT_02052]	Definition of Rapid Prototyping Scenario is splittable
[TPS_SWCT_02053]	Values of RuleBasedAxisCont with the <i>category</i> CURVE_AXIS, COM_AXIS, RES_AXIS are for display only
[TPS_SWCT_02507]	Instantiation-specific RTEEvents

Table C.12: Added Specification Items in 4.1.1

C.4.5 Deleted Constraints in R4.1.1

Number	Heading
[constr_1239]	RuleBasedValueSpecification shall not exceed the number of values required
[constr_1145]	Finding the symbol for the representation of a CompuScale in C code
[constr_2533]	Iteration along output axis is only supported for VALUE and VAL_BLK

Table C.13: Deleted Constraints in R4.1.1

Please note that [constr_2533] has been retagged to [[constr_1264](#)] to fix a duplicate constraint ID.

C.4.6 Deleted Specification Items in R4.1.1

Number	Heading
[TPS_SWCT_01210]	IoHwAbstractionServerAnnotation

Table C.14: Deleted Specification Items in R4.1.1

C.5 Constraint History of this Document according to AUTOSAR R4.1.2

C.5.1 Changed Constraints in R4.1.2

Number	Heading
[constr_1006]	applicable data categories
[constr_1007]	Allowed attributes of <code>SwDataDefProps</code> for <code>ApplicationDataTypes</code>
[constr_1009]	<code>SwDataDefProps</code> applicable to <code>ImplementationDataTypes</code>
[constr_1015]	Prioritization of <code>SwDataDefProps</code>
[constr_1043]	<code>PortInterface</code> vs. <code>ComSpec</code>
[constr_1058]	<code>ImplementationDataType</code> has category <code>FUNCTION_REFERENCE</code>
[constr_1102]	<code>ApplicationError</code> in the scope of one <code>SwComponentType</code>
[constr_1146]	Applicability of a symbol for a <code>CompuScale</code> in C code
[constr_1163]	Compatibility of <code>CompuMethods</code>
[constr_1133]	Identical <code>CompuScale</code> Symbolic Names shall have the same range
[constr_1158]	Applicable categories for attribute <code>ImplementationDataType.swDataDefProps.compuMethod</code>
[constr_1225]	<code>DataPrototype</code> is typed by an <code>ImplementationDataType</code> that references a <code>CompuMethod</code> of category <code>TEXTTABLE</code> or <code>BITFIELD_TEXTTABLE</code>
[constr_2013]	Compatibility of <code>ImplementationDataTypes</code> for <code>NvBlockDataMapping</code>
[constr_2051]	Mandatory information of a <code>SwValueCont</code>

Table C.15: Changed Constraints in R4.1.2

C.5.2 Added Constraints in R4.1.2

Number	Heading
[constr_1277]	<code>SwDataDefProps.swImplPolicy</code> of a <code>VariableDataPrototype</code> referenced by a <code>VariableAccess</code> aggregated in the role <code>dataReceivePointByValue</code>
[constr_1278]	<code>PhysConstrs</code> references a <code>Unit</code>
[constr_1279]	Unmapped elements of <code>ApplicationCompositeDataTypes</code> or <code>ImplementationDataTypes</code> and the attribute <code>swImplPolicy</code>
[constr_1280]	Unmapped <code>dataElement</code> on the receiver side shall have an <code>initValue</code>
[constr_1281]	<code>invalidValue</code> shall be inside the scope of the <code>compuMethod</code>
[constr_1282]	Restriction concerning the usage of <code>RuleBasedValueSpecification</code> or a <code>ReferenceValueSpecification</code> for the specification of an <code>invalidValue</code>
[constr_1283]	<code>invalidValue</code> is outside the scope of the <code>compuMethod</code>
[constr_1284]	Limitation of the use of <code>TextValueSpecification</code>
[constr_1285]	Applicability of roles vs. <code>PortPrototypes</code>
[constr_1286]	<code>serverArgumentImplPolicy</code> and <code>ArgumentDataPrototype</code> typed by primitive data types
[constr_1287]	Compatibility of <code>SenderReceiverInterfaces</code> with respect to <code>invalidationPolicy</code>
[constr_1288]	Allowed Attributes vs. category for <code>DataPrototypes</code> typed by <code>ImplementationDataTypes</code>
[constr_1289]	Allowed Attributes vs. category for <code>DataPrototypes</code> typed by <code>ApplicationDataTypes</code>
[constr_1290]	Limitation on the number of <code>PPortComSpecs</code> in the context of one <code>PPortPrototype</code>
[constr_1291]	Limitation on the number of <code>RPortComSpecs</code> in the context of one <code>PPortPrototype</code>
[constr_1292]	Limitation on the number of <code>RPortComSpecs/PPortComSpecs</code> in the context of one <code>PRPortPrototype</code>
[constr_1293]	Existence of <code>DiagnosticEventNeeds.dtcNumber</code>
[constr_1294]	Existence of <code>DiagnosticEventInfoNeeds.dtcNumber</code>
[constr_1295]	<code>PortInterfaces</code> and category <code>DATA_REFERENCE</code>
[constr_1296]	<code>DataPrototypes</code> used as <code>explicitInterRunnableVariable</code> or <code>implicitInterRunnableVariable</code> and category <code>DATA_REFERENCE</code>

Table C.16: Added Constraints in R4.1.2

C.5.3 Changed Specification Items in R4.1.2

Number	Heading
[TPS_SWCT_01006]	ImplementationDataType.subElement.arraySize shall be used to define the size of the array
[TPS_SWCT_01175]	Actively query the status of a partial network
[TPS_SWCT_01219]	ComSpec for queued and non-queued sender-receiver communication
[TPS_SWCT_01154]	Attribute prefix of IncludedModeDeclarationGroupSet
[TPS_SWCT_02011]	AtomicSwComponentType offers a server port to read DTR value via OBD services

Table C.17: Changed Specification Items in R4.1.2

C.5.4 Added Specification Items in R4.1.2

Number	Heading
[TPS_SWCT_01549]	Definition of linear data scaling
[TPS_SWCT_01550]	Definition of reciprocal linear data scaling
[TPS_SWCT_01551]	Mapping of elements on the sender side to elements on the receiver side
[TPS_SWCT_01552]	Software-component acts as a mode manager
[TPS_SWCT_01553]	Software-component acts as a mode user
[TPS_SWCT_01554]	Software-component acts as a mode requester
[TPS_SWCT_01555]	ModeSwitchedAckEvent is triggered by the RTE regardless
[TPS_SWCT_01556]	Rule for setting RoleBasedPortAssignment.role
[TPS_SWCT_01557]	dataWriteAccess also allows for the definition of a DataWriteCompletedEvent
[TPS_SWCT_01558]	DataWriteCompletedEvent cannot be combined with a WaitPoint
[TPS_SWCT_01559]	Default value for attribute SwDataDefProps.swCalibrationAccess
[TPS_SWCT_01560]	Supported categorys of CompuMethods for data conversion
[TPS_SWCT_01561]	Application of data conversion to composite AutosarDataTypes
[TPS_SWCT_01562]	Specification of values of an enumeration
[TPS_SWCT_01563]	Applicable values for nativeDeclaration
[TPS_SWCT_01564]	Non-recursive definition of a primitive data type
[TPS_SWCT_01565]	Recursive definition of a primitive data type
[TPS_SWCT_01566]	Define literals for an MCD system in the context of a FlatInstanceDescriptor
[TPS_SWCT_01567]	Default behavior for invalidationPolicy
[TPS_SWCT_01568]	Consideration of RPortComSpec or PPortComSpec depending on the ownership
[TPS_SWCT_01569]	Definition of CompuScale Symbolic Name
[TPS_SWCT_01570]	DataTypeMap is mandatory in the presence of ApplicationPrimitive-DataType.swDataDefProps.swRecordLayout

Table C.18: Added Specification Items in 4.1.2

C.5.5 Deleted Constraints in R4.1.2

Number	Heading
[constr_1042]	Definition of a linear data scaling
[constr_1094]	Usage of symbol of RunnableEntity
[constr_2025]	Uniqueness of symbol attributes (This constraint has been relocated to [11])
[constr_2032]	transmissionAcknowledge requires a DataSendCompletedEvent
[constr_4011]	ComSpec and ModeSwitchedAckEvent

Table C.19: Deleted Constraints in R4.1.2

C.5.6 Deleted Specification Items in R4.1.2

Number	Heading
[TPS_SWCT_01327]	RTE generator can omit the creation of checks at run-time

Table C.20: Deleted Specification Items in R4.1.2

C.6 Constraint History of this Document according to AUTOSAR R4.1.3

C.6.1 Added Traceables in 4.1.3

Id	Heading
[TPS_SWCT_01573]	A PRPortPrototype is never considered unconnected
[TPS_SWCT_01574]	PerInstanceMemory.typeDefinition shall not contain a function pointer

Table C.21: Added Traceables in 4.1.3

C.6.2 Changed Traceables in 4.1.3

Id	Heading
[TPS_SWCT_01009]	The numerical values used to define the values of ModeDeclaration.value and ModeDeclarationGroup.onTransitionValue can be arbitrarily defined
[TPS_SWCT_01049]	Two ways to use the ExclusiveAreas

Table C.22: Changed Traceables in 4.1.3

C.6.3 Deleted Traceables in 4.1.3

Id	Heading
[TPS_SWCT_01417]	Define calibration parameters common to all SwComponentPrototypes of the same SwComponentType
[TPS_SWCT_01419]	ParameterSwComponentType shall never aggregate a SwcInternalBehavior
[TPS_SWCT_02006]	AtomicSwComponentType offers sender receiver ports to adjust the IO signal via diagnostic services

Table C.23: Deleted Traceables in 4.1.3

C.6.4 Added Constraints in 4.1.3

Id	Heading
[constr_1297]	Applicability of serverArgumentImplPolicy set to useArrayBaseType
[constr_1298]	Existence of attributes if category of a ModeDeclarationGroup is set to EXPLICIT_ORDER
[constr_1299]	Existence of attributes if category of a ModeDeclarationGroup is set to other than EXPLICIT_ORDER

Table C.24: Added Constraints in 4.1.3

C.6.5 Changed Constraints in 4.1.3

Id	Heading
[constr_1006]	applicable data categories
[constr_1007]	Allowed attributes of SwDataDefProps for ApplicationDataTypes
[constr_1230]	ApplicationDataType that qualifies for Integral Primitive Type

Table C.25: Changed Constraints in 4.1.3

C.6.6 Deleted Constraints in 4.1.3

Id	Heading
[constr_1179]	Existence of ModeDeclaration.value within a ModeDeclarationGroup
[constr_1180]	Existence of ModeDeclarationGroup.onTransitionValue

Table C.26: Deleted Constraints in 4.1.3

C.7 Constraint History of this Document according to AUTOSAR R4.2.1

C.7.1 Added Traceables in 4.2.1

Id	Heading
[TPS_SWCT_01572]	Application Mode Manager interacts with both BswM and other ApplicationSwComponentTypes
[TPS_SWCT_01577]	AtomicSwComponentType requires the notification about a Service Request via diagnostic services with <i>manufacturer</i> characteristics
[TPS_SWCT_01578]	AtomicSwComponentType requires the notification about a Service Request via diagnostic services with <i>supplier</i> characteristics
[TPS_SWCT_01579]	Dcm can directly access dataElements in PPortPrototypes typed by a SenderReceiverInterface
[TPS_SWCT_01580]	Dem can directly access dataElements in PPortPrototypes typed by a SenderReceiverInterface
[TPS_SWCT_01581]	Communication patterns for mode-related communication
[TPS_SWCT_01582]	Semantics of DiagnosticEventNeeds.deferringFid
[TPS_SWCT_01583]	Completeness of TextTableMapping is not a requirement
[TPS_SWCT_01584]	InternalBehavior of a NvBlockSwComponentType for implementing a writing strategy

[TPS_SWCT_01585]	Relevance of <code>NvBlockDescriptor.timingEvent.period</code>
[TPS_SWCT_01586]	Writing strategies for <i>nv data</i>
[TPS_SWCT_01587]	The cyclic writing of <i>nv data</i> requires the existence of a <code>TimingEvent</code>
[TPS_SWCT_01588]	<code>DataReceivedEvent</code> for storing <i>nv data</i> immediately
[TPS_SWCT_01589]	Implementation of emergency storing of <i>nv data</i>
[TPS_SWCT_01590]	Combination of writing strategies for <i>nv data</i> is possible
[TPS_SWCT_01591]	Existence of attribute <code>DiagnosticEventNeeds.reportBehavior</code>
[TPS_SWCT_01592]	Communication among <code>RunnableEntitys</code> of different instances of the same <code>AtomicSwComponentType</code>
[TPS_SWCT_01593]	Semantics of attribute <code>ReceiverComSpec.transformationComSpecProps</code>
[TPS_SWCT_01594]	Semantics of <code>TransformationComSpecProps</code>
[TPS_SWCT_01595]	Semantics of attribute <code>ClientComSpec.transformationComSpecProps</code>
[TPS_SWCT_01596]	Semantics of attribute <code>ServerComSpec.transformationComSpecProps</code>
[TPS_SWCT_01597]	<code>PortPrototype</code> -specific data transformation configuration
[TPS_SWCT_01598]	More than one user-defined transformer is used within one transformer chain
[TPS_SWCT_01599]	<code>PortPrototype</code> -specific configuration for custom transformers
[TPS_SWCT_01600]	<code>PortPrototype</code> -specific configuration for data transformers related to end-to-end protection
[TPS_SWCT_01601]	<code>Size Indicator</code> shall be updated by software-component
[TPS_SWCT_01602]	<code>Size Indicator</code> shall be read by the software-component
[TPS_SWCT_01603]	Variable size array with <code>Size Indicator</code>
[TPS_SWCT_01604]	Enable <code>Size Indicator</code>
[TPS_SWCT_01605]	Semantics of <code>ApplicationArrayElement.arraySizeHandling</code>
[TPS_SWCT_01606]	Internal structure of mapped <code>ImplementationDataType</code>
[TPS_SWCT_01607]	Profiles for internal structure of mapped <code>ImplementationDataType</code>
[TPS_SWCT_01608]	Custom profiles for internal structure of mapped <code>ImplementationDataType</code>
[TPS_SWCT_01609]	A <code>RuleBasedValueSpecification</code> of rule <code>FILL_UNTIL_MAX_SIZE</code> shall fill the value of the last <code>RuleArguments.argument</code> until the number of elements specified in <code>maxSizeToFill</code>
[TPS_SWCT_01610]	Modeling of a <code>Variable-Size Array Data Type</code> with <code>Size Indicator</code> enabled
[TPS_SWCT_01611]	Enable <code>Size Indicator</code>
[TPS_SWCT_01612]	<code>arraySizeHandling</code> specifies how the size is determined
[TPS_SWCT_01613]	Internal structure of mapped <code>ImplementationDataType</code>
[TPS_SWCT_01614]	Profiles for internal structure of mapped <code>ImplementationDataType</code>
[TPS_SWCT_01615]	Custom profiles for internal structure of mapped <code>ImplementationDataType</code>
[TPS_SWCT_01616]	Semantics of <code>TransformerHardErrorEvent</code>
[TPS_SWCT_01617]	Structure of an <code>ImplementationDataType</code> that represents a variable-sized array data type
[TPS_SWCT_01618]	<code>Size Indicator</code> for <code>dynamicArraySizeProfile</code> set to <code>VSA_LINEAR</code> , <code>VSA_SQUARE</code> , or <code>VSA_FULLY_FLEXIBLE</code>
[TPS_SWCT_01619]	<code>Size Indicator</code> for <code>dynamicArraySizeProfile</code> set to <code>VSA_RECTANGULAR</code>
[TPS_SWCT_01620]	<code>Size Indicator</code> for <code>dynamicArraySizeProfile</code> set to <code>VSA_RECTANGULAR</code>
[TPS_SWCT_01621]	Payload for <code>dynamicArraySizeProfile</code>
[TPS_SWCT_01622]	Modeling of a <code>Variable-Size Array Data Type</code> only with <code>ImplementationDataType</code>

[TPS_SWCT_01623]	Justification for the existence of attributes <code>ApplicationArray-DataType.dynamicArraySizeProfile</code> and <code>ApplicationArrayElement.arraySizeHandling</code>
[TPS_SWCT_01624]	Hard error occurs during the execution of a transformer chain
[TPS_SWCT_01625]	Sending <code>SwComponentType</code> owns a <code>DataPrototypeGroup</code> in the role <code>dp-gRequiresCoherence</code> and also <code>RunnableEntityGroups</code>
[TPS_SWCT_01626]	Error notification of data transformer errors
[TPS_SWCT_01627]	Suffix used for the resulting name of the <code>PortInterface</code> for the Security Access
[TPS_SWCT_01628]	Suffix used for the resulting name of the <code>PortInterface</code> for the Data Services
[TPS_SWCT_01629]	Suffix used for the resulting name of the <code>PortInterface</code> for the Data Services
[TPS_SWCT_01630]	Suffix used for the resulting name of the <code>PortInterface</code> for the Data Services
[TPS_SWCT_01631]	Suffix used for the resulting name of the <code>PortInterface</code> for the Infotype Services
[TPS_SWCT_01632]	Suffix used for the resulting name of the <code>PortInterface</code> for the Routine Services
[TPS_SWCT_01633]	Suffix used for the resulting name of the <code>PortInterface</code> for the Request Control Services
[TPS_SWCT_01634]	Suffix used for the resulting name of the <code>PortInterface</code> for the Data Services
[TPS_SWCT_01635]	Naming conventions may support the effectiveness of <code>SymbolProps</code>
[TPS_SWCT_01636]	Definition of profiles for the definition of <code>Variable-Size Array Data Types</code>

Table C.27: Added Traceables in 4.2.1

C.7.2 Changed Traceables in 4.2.1

Id	Heading
[TPS_SWCT_01044]	<code>ServiceNeeds</code>
[TPS_SWCT_01053]	Relationship of interchanged data with <code>RunnableEntitys</code>
[TPS_SWCT_01141]	<code>AtomicSwComponentType</code> may have <code>AbstractRequiredPortPrototypes</code> typed by an <code>NvDataInterface</code>
[TPS_SWCT_01150]	<code>InternalBehavior</code> of a <code>NvBlockSwComponentType</code> to enable access to the NVRAM Block management API
[TPS_SWCT_01162]	Existence of <code>TextTableMapping</code>
[TPS_SWCT_01180]	Maximum possible size of Compound Primitive Data Type
[TPS_SWCT_01227]	Unconnected <code>AbstractRequiredPortPrototype</code> typed by <code>NvDataInterface</code>
[TPS_SWCT_01228]	<code>NvProvideComSpec</code>
[TPS_SWCT_01239]	default value for attribute <code>category</code> used in the context of <code>SwSystemconst</code>
[TPS_SWCT_01274]	<code>SwDataDefProps</code> used to support calibration and measurement
[TPS_SWCT_01321]	Communication among <code>RunnableEntitys</code>
[TPS_SWCT_01330]	<code>RunnableEntity</code> can also have <code>dataSendPoints</code>
[TPS_SWCT_01335]	Combine <code>dataReceivePointByValue</code> or <code>dataReceivePointByArgument</code> with a <code>WaitPoint</code>
[TPS_SWCT_01348]	Trigger source
[TPS_SWCT_01385]	Execution of finalization code for software-components
[TPS_SWCT_01456]	Predefined values for <code>MemorySection.option</code> and <code>SwAddrMethod.option</code>

[TPS_SWCT_01475]	Sending <code>SwComponentType</code> owns a <code>DataPrototypeGroup</code> in the role <code>dp-gRequiresCoherence</code>
[TPS_SWCT_01494]	A <code>RuleBasedValueSpecification</code> of rule <code>FILL_UNTIL_END</code> shall fill the value of the last <code>RuleArguments.argument</code> until the last element of the array
[TPS_SWCT_01495]	Standardized value of <code>RuleBasedValueSpecification.rule</code>
[TPS_SWCT_01549]	Definition of linear data scaling
[TPS_SWCT_01560]	Supported <code>categorys</code> of <code>CompuMethods</code> for data conversion
[TPS_SWCT_02501]	Setup for Nvm Use Case: Permanent RAM Block
[TPS_SWCT_02502]	Setup for Nvm Use Case: Temporary RAM Block
[TPS_SWCT_02503]	Setup for NVM Use Case: Software-Components using Nv Data provided by <code>NvBlockSwComponentType</code>
[TPS_SWCT_02504]	Setup for Nvm Use Case: RAM Block with explicit synchronization using Mirror Interfaces

Table C.28: Changed Traceables in 4.2.1

C.7.3 Deleted Traceables in 4.2.1

Id	Heading
[TPS_SWCT_01131]	<code>AtomicSwComponentType</code> accepts a request to restart an entire function
[TPS_SWCT_01386]	Initialization by mode management
[TPS_SWCT_01387]	Finalization by mode management
[TPS_SWCT_01410]	Dcm and Dem can directly access <code>dataElements</code> in <code>PPortPrototypes</code> typed by a <code>SenderReceiverInterface</code>
[TPS_SWCT_01438]	Handling of invalidation in the sending RTE
[TPS_SWCT_01439]	Handling of invalidation in the receiving RTE
[TPS_SWCT_02017]	<code>AtomicSwComponentType</code> requires the notification about a Service Request via diagnostic services

Table C.29: Deleted Traceables in 4.2.1

C.7.4 Added Constraints in 4.2.1

Id	Heading
[constr_1300]	Primitive <code>DataPrototype</code> on the provider side shall not be mapped to element of a composite data type on the requester side
[constr_1301]	Existence of <code>RoleBasedDataTypeAssignment.role</code> vs. <code>RoleBasedDataAssignment.role</code>
[constr_1302]	Restriction of data invalidation
[constr_1303]	Applicability of <code>TextTableMapping</code> depending on the value of <code>CompuMethod.category</code>
[constr_1304]	Existence of attribute <code>bitfieldTextTableMaskFirst</code>
[constr_1305]	Existence of attribute <code>bitfieldTextTableMaskSecond</code>
[constr_1306]	Limitation of <code>TextTableMapping</code> for <code>CompuMethods</code> that have the value of <code>category</code> set to <code>BITFIELD_TEXTTABLE</code>
[constr_1307]	Consistency of values and masks in <code>TextTableMapping</code>
[constr_1308]	Existence of <code>NvBlockNeeds.cyclicWritingPeriod</code>
[constr_1309]	Existence of <code>NvBlockDescriptor.timingEvent</code>
[constr_1310]	Existence of attributes of meta-class <code>NvBlockNeeds</code>
[constr_1311]	Appearance of safety-related possible values of <code>MemorySection.option</code> or <code>SwAddrMethod.option</code> according to [TPS_SWCT_01456]

[constr_1312]	PortPrototypes typed by a ParameterInterface
[constr_1313]	Completeness of TextTableMapping for the values of a given bit mask on the sender side
[constr_1314]	Profile VSA_LINEAR for ApplicationArrayDataType
[constr_1315]	Profile VSA_SQUARE for ApplicationArrayDataType
[constr_1316]	Profile VSA_RECTANGULAR for ApplicationArrayDataType
[constr_1317]	Profile VSA_FULLY_FLEXIBLE for ApplicationArrayDataType
[constr_1318]	Profile VSA_LINEAR for ImplementationDataType
[constr_1319]	Profile VSA_SQUARE for ImplementationDataType
[constr_1320]	Profile VSA_RECTANGULAR for ImplementationDataType
[constr_1321]	Profile VSA_FULLY_FLEXIBLE for ImplementationDataType
[constr_1322]	Size Indicator for undefined dynamicArraySizeProfile
[constr_1323]	Applicability of attribute ReceiverComSpec.usesEndToEndProtection
[constr_1363]	Existence of attributes of DiagnosticValueNeeds
[constr_1364]	Existence of attributes of DiagnosticToControlNeeds
[constr_1375]	Existence of attributes of CompuMethod and related meta-classes

Table C.30: Added Constraints in 4.2.1

C.7.5 Changed Constraints in 4.2.1

Id	Heading
[constr_1006]	applicable data categories
[constr_1007]	Allowed attributes of SwDataDefProps for ApplicationDataTypes
[constr_1009]	SwDataDefProps applicable to ImplementationDataTypes
[constr_1014]	Supported value encodings for SwBaseType
[constr_1015]	Prioritization of SwDataDefProps
[constr_1037]	Client shall not be connected to multiple servers
[constr_1072]	Compatibility of ModeSwitchInterfaces in the context of an AssemblySwConnector
[constr_1073]	Compatibility of ModeSwitchInterfaces in the context of an Delegation-SwConnector
[constr_1074]	Compatibility of ModeDeclarationGroupPrototypes
[constr_1101]	Mode-related communication
[constr_1112]	Constraints of dataIdMode in PROFILE_01
[constr_1149]	PortPrototypes used for NV data management
[constr_1188]	Existence of ReceiverComSpec.replaceWith
[constr_1288]	Allowed Attributes vs. category for DataPrototypes typed by ImplementationDataTypes
[constr_1289]	Allowed Attributes vs. category for DataPrototypes typed by ApplicationDataTypes
[constr_2009]	Supported kinds of PortPrototypes of a NvBlockSwComponentType
[constr_2015]	Limitation of SwcInternalBehavior of a NvBlockSwComponentType
[constr_2019]	ServiceSwComponentType shall have service ports only
[constr_2022]	Mutually exclusive use of SynchronousServerCallPoints and AsynchronousServerCallPoints
[constr_2051]	Mandatory information of a SwValueCont

Table C.31: Changed Constraints in 4.2.1

C.7.6 Deleted Constraints in 4.2.1

Id	Heading
[constr_1189]	Allowed targets of externalReplacement
[constr_1293]	Existence of DiagnosticEventNeeds.dtcNumber
[constr_1294]	Existence of DiagnosticEventInfoNeeds.dtcNumber
[constr_2551]	SwCalprmAxis.baseType shall be ignored

Table C.32: Deleted Constraints in 4.2.1

C.8 Constraint History of this Document according to AUTOSAR R4.2.2

C.8.1 Added Traceables in 4.2.2

Id	Heading
[TPS_SWCT_01637]	Initial value for a specific implicitInterRunnableVariable or explicitInterRunnableVariable
[TPS_SWCT_01638]	Existence of SwConnector between two PRPortPrototypes
[TPS_SWCT_01639]	AtomicSwComponentType offers a PPortPrototype typed by ClientServerInterface to read/write current value via diagnostic services where the applicable DID is passed as an argument to the access functions
[TPS_SWCT_01640]	Suffix used for the resulting name of the PortInterface for the Data Services
[TPS_SWCT_01641]	Definition of an “old-world” dynamic-size array data type by means of an ApplicationArrayDataType
[TPS_SWCT_01642]	Definition of an “old-world” dynamic-size array data type by means of an ImplementationDataType
[TPS_SWCT_01644]	Definition of a “new-world” variable-size array data type by means of an ApplicationArrayDataType
[TPS_SWCT_01645]	Definition of a “new-world” variable-size array data type by means of an ImplementationDataType
[TPS_SWCT_01646]	Sending invalidValue without invalidation applied by RTE/Com
[TPS_SWCT_01647]	Size Indicator for dynamicArraySizeProfile set to VSA_LINEAR , VSA_SQUARE , or VSA_FULLY_FLEXIBLE if only ImplementationDataType is present
[TPS_SWCT_01648]	Size Indicator for dynamicArraySizeProfile set to VSA_RECTANGULAR if only ImplementationDataType is present
[TPS_SWCT_01649]	Payload for dynamicArraySizeProfile if only ImplementationDataType is present
[TPS_SWCT_01650]	Structure of the VSA ImplementationDataType
[TPS_SWCT_01651]	UTF-16BE
[TPS_SWCT_01652]	UTF-16LE
[TPS_SWCT_01653]	UTF-16-encoded strings are not allowed to start with a BOM
[TPS_SWCT_01654]	AtomicSwComponentType offers PortPrototypes typed by SenderReceiverInterfaces to adjust the IO signal via diagnostic services
[TPS_SWCT_01655]	Reference from DiagnosticIoControlNeeds to DiagnosticValueNeeds
[TPS_SWCT_01656]	Suffix used for the resulting name of the PortInterface for IOControlRequest and IOControlResponse
[TPS_SWCT_01657]	NamingRule for RPortPrototype referenced by a RoleBasedPortAssignment with attribute role set to “ IOControlRequest ”

[TPS_SWCT_01658]	NamingRule for <code>PPortPrototype</code> referenced by a <code>RoleBasedPortAssignment</code> with attribute <code>role</code> set to "IOControlResponse"
[TPS_SWCT_01659]	Mapping of <code>VariableDataPrototype</code> to a <code>NvBlockDescriptor</code>

Table C.33: Added Traceables in 4.2.2

C.8.2 Changed Traceables in 4.2.2

Id	Heading
[TPS_SWCT_01133]	<code>AtomicSwComponentType</code> provides information about aging cycles
[TPS_SWCT_01158]	Three cases for <code>PortInterfaceMapping</code>
[TPS_SWCT_01179]	Compound Primitive Data Type
[TPS_SWCT_01244]	Data types for calibration parameters are also described as primitive types
[TPS_SWCT_01370]	<code>VariationPointProxy</code>
[TPS_SWCT_01431]	Finding the symbol for the representation of a <code>CompuScale</code> with a point-range in C code
[TPS_SWCT_01432]	Keep the <code>invalidValue</code> transparent to the sending and receiving software components
[TPS_SWCT_01434]	Sender and receiver have knowledge of invalid value
[TPS_SWCT_01610]	Modeling of a <code>Variable-Size Array Data Type</code> with <code>Size Indicator</code> enabled
[TPS_SWCT_01616]	Semantics of <code>TransformerHardErrorEvent</code>
[TPS_SWCT_01617]	Structure of an <code>ImplementationDataType</code> that represents a variable-sized array data type
[TPS_SWCT_01624]	Hard error occurs during the execution of a transformer chain
[TPS_SWCT_02001]	Values of <code>SwAxisCont</code> with the <code>category</code> <code>CURVE_AXIS</code> , <code>COM_AXIS</code> , <code>RES_AXIS</code> are for display only
[TPS_SWCT_02053]	Values of <code>RuleBasedAxisCont</code> with the <code>category</code> <code>CURVE_AXIS</code> , <code>COM_AXIS</code> , <code>RES_AXIS</code> are for display only

Table C.34: Changed Traceables in 4.2.2

C.8.3 Deleted Traceables in 4.2.2

Id	Heading
[TPS_SWCT_01308]	Combination of <code>supportsMultipleInstantiation=false</code> and <code>CanBeInvokedConcurrently=false</code>
[TPS_SWCT_01433]	Invalid values outside the range limits
[TPS_SWCT_01435]	Invalid values outside the range limits
[TPS_SWCT_01603]	Variable size array with <code>Size Indicator</code>
[TPS_SWCT_01611]	Enable <code>Size Indicator</code>

Table C.35: Deleted Traceables in 4.2.2

C.8.4 Added Constraints in 4.2.2

Id	Heading
[constr_1381]	Appearance of core-related possible values of <code>MemorySection.option</code> or <code>SwAddrMethod.option</code> according to [TPS_SWCT_01456]
[constr_1382]	Mutually exclusive existence of attributes <code>SwVariableRefProxy.autosarVariable</code> vs. <code>SwVariableRefProxy.mcDataInstanceVar</code>

[constr_1383]	Existence of <code>CompuMethod</code> and <code>DataConstr</code> for <code>ImplementationDataTypes</code> of category <code>TYPE_REFERENCE</code>
[constr_1384]	Definition of <code>invalidValue</code> for <code>DataPrototype</code> typed by <code>ApplicationPrimitiveDataType</code> of category <code>CURVE</code> , <code>MAP</code> , <code>CUBOID</code> , <code>CUBE_4</code> , <code>CUBE_5</code> , <code>COM_AXIS</code> , <code>RES_AXIS</code> , and <code>VAL_BLK</code>
[constr_1385]	<code>DataPrototype</code> is typed by an <code>ImplementationDataType</code>
[constr_1386]	<code>PortDefinedArgumentValue</code> shall only be defined for <code>AbstractProvidedPortPrototype</code>
[constr_1388]	<code>VariationPointProxy</code> of category <code>VALUE</code> shall not mix “pre-build” and “post-build” use-cases
[constr_1389]	Restriction regarding the value of <code>category</code> of <code>VariationPointProxy.implementationDataType</code>
[constr_1390]	Restriction to the value of <code>SenderReceiverInterface.invalidationPolicy.handleInvalid</code>
[constr_1391]	Compatibility of <code>Units</code> in the context of assignment using an <code>ApplicationValueSpecification</code>
[constr_1392]	Compatibility of <code>Units</code> in the context of assignment using an <code>ApplicationRuleBasedValueSpecification</code>
[constr_1393]	Existence of <code>RuleBasedValueCont.unit</code>
[constr_1395]	<code>NvBlockDataMapping</code> shall be complete
[constr_1396]	Restriction for the value of attribute <code>category</code> for non-terminating <code>ImplementationDataTypeElements</code> taken to model a <code>Variable-Size Array Data Type</code>
[constr_1397]	Existence of attributes of <code>TransformerHardErrorEvent</code>
[constr_1398]	Existence of attributes of <code>BaseTypeDirectDefinition</code>
[constr_1399]	Standardized values of <code>ModeDeclarationGroup.category</code>
[constr_1400]	Reference to a specific <code>DataTransformation</code>
[constr_1401]	Restrictions on the relation between <code>DataPrototypeMapping</code> and <code>DataTransformation</code>
[constr_1402]	Applicability of core-related possible values of <code>MemorySection.option</code> or <code>SwAddrMethod.option</code> related to <code>SwAddrMethod.sectionInitializationPolicy</code>
[constr_1403]	<code>NvBlockDataMappings</code> to a given <code>nvData</code> shall be unambiguous
[constr_1404]	All <code>NvDataInterface.nvData</code> of <code>PortPrototypes</code> in the context of a specific <code>SwcServiceDependency</code> shall be mapped to the same <code>NvBlockDescriptor</code>

Table C.36: Added Constraints in 4.2.2

C.8.5 Changed Constraints in 4.2.2

Id	Heading
[constr_1006]	applicable data categories
[constr_1007]	Allowed attributes of <code>SwDataDefProps</code> for <code>ApplicationDataTypes</code>
[constr_1040]	Conversion of <code>SenderReceiverInterfaces</code>
[constr_1045]	Supported value encodings for <code>SwBaseType</code> in the context of <code>PortInterfaces</code>
[constr_1060]	Compatibility of data types with category <code>ARRAY</code> , <code>VAL_BLK</code>
[constr_1064]	Compatibility of data types with category <code>COM_AXIS</code> , <code>RES_AXIS</code> , <code>CURVE</code> , <code>MAP</code> , <code>CUBOID</code> , <code>CUBE_4</code> , or <code>CUBE_5</code>
[constr_1066]	Forbidden mappings to <code>ImplementationDataType</code>
[constr_1075]	Compatibility of <code>ModeDeclarationGroups</code>
[constr_1164]	Number of <code>arguments</code> owned by a <code>RunnableEntity</code>
[constr_1234]	Value of <code>RunnableEntity.symbol</code>
[constr_1318]	Profile <code>VSA_LINEAR</code> for <code>ImplementationDataType</code>
[constr_1319]	Profile <code>VSA_SQUARE</code> for <code>ImplementationDataType</code>
[constr_1320]	Profile <code>VSA_RECTANGULAR</code> for <code>ImplementationDataType</code>

[constr_1321]	Profile VSA_FULLY_FLEXIBLE for ImplementationDataType
[constr_1322]	Size Indicator for undefined dynamicArraySizeProfile
[constr_2051]	Mandatory information of a SwValueCont
[constr_2058]	Mandatory information of a RuleBasedValueCont

Table C.37: Changed Constraints in 4.2.2

C.8.6 Deleted Constraints in 4.2.2

Id	Heading
[constr_1002]	End-to-end protection does not support n:1 communication
[constr_1067]	ApplicationDataType is or is not compatible to specific Implementation-DataType
[constr_2001]	Initial value for a specific implicitInterRunnableVariable or explicitInterRunnableVariable

Table C.38: Deleted Constraints in 4.2.2

D Modeling of InstanceRef

D.1 Introduction

The existence of so-called `InstanceRefs` is a direct consequence to the usage of the `type-prototype` pattern for modeling within AUTOSAR. When referencing a `prototype` it is also necessary to include a reference to the `prototypes` typed by their corresponding `types` that in turn aggregate further `prototypes` to set up the context.

In other words, `InstanceRefs` are representing **structured references** that, on the one hand, consist of references to context `prototypes` (indicated by a subsetting or redefinition of `atpContextElement`) and finally a reference to the applicable target `prototype` (indicated by a redefinition of `atpTarget`).

Note that it is not uncommon to have more than a single context in the modeling of particular `InstanceRefs`.

For the reader of specifications, the modeling of `InstanceRefs` manifests as a UML dependency stereotyped `<<instanceRef>>` drawn from one meta-class to another. This is a simplified indication that the source of the dependency implements an `InstanceRef` to the meta-class at the target of the dependency. Again, in most cases this is everything a reader needs to understand in order to figure out the modeling. The formal modeling of `InstanceRefs` is done by creating subclasses of the abstract meta-class `AtpInstanceRef`.

Wherever a more detailed understanding of the modeling is advised in the context of the specific chapter of this document, the modeling of a specific subclasses of `AtpInstanceRef` is explained directly in the context of the corresponding chapter. In all other cases, a deeper understanding of the modeling of particular subclasses of `AtpInstanceRefs` can be obtained from reading this chapter.

Class tables included in this chapter are not fully filled out in the sense that most of the notes inside the class tables are missing. The **primary** purpose of these class tables is to **provide information about the intended order in which `InstanceRefs` are serialized in M1 AUTOSAR models**.

In particular, the information about the order in serialized M1 models can be obtained from the value of the tag `xml.sequenceOffset` of each attribute of an `InstanceRef` meta-class.

For more information about the general concept of modeling `AtpInstanceRef` (e.g. the conceptual background of redefining or subsetting an association from a subclass of `AtpInstanceRef` to other meta-classes) please refer to [12].

D.2 Modeling

D.2.1 Components and Compositions

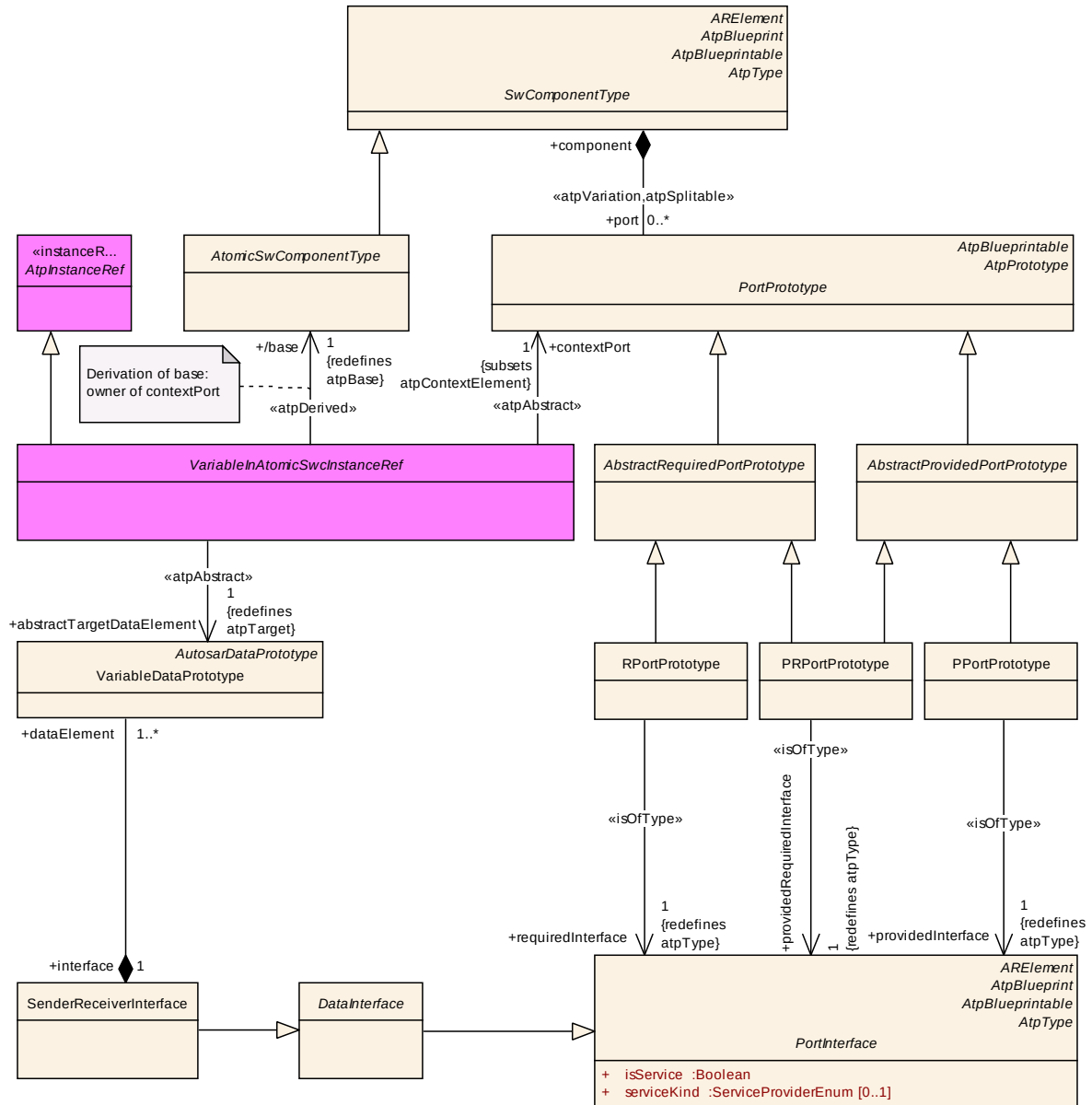


Figure D.1: Abstract modeling of references to **VariableDataPrototype** in the context of a **AtomicSwComponentType**

Class	VariableInAtomicSwcInstanceRef (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components::InstanceRefs			
Note				
Base	ARObject, AtpInstanceRef			
Attribute	Datatype	Mul.	Kind	Note

Attribute	Datatype	Mul.	Kind	Note
abstractTargetDataElement	VariableDataPrototype	1	ref	Stereotypes: atpAbstract Tags: xml.sequence Offset=30
base	AtomicSwComponentType	1	ref	Stereotypes: atpDerived Tags: xml.sequence Offset=10
contextPort	PortPrototype	1	ref	Stereotypes: atpAbstract Tags: xml.sequence Offset=20

Table D.1: VariableInAtomicSwcInstanceRef

Please note the example of how the redefinition of the context association works, i.e. the association from `VariableInAtomicSwcInstanceRef` to `PortPrototype` in the role `contextPort` is **redefined** by the subclass `RVariableInAtomicSwcInstanceRef` by means of an association to `RPortPrototype` in the role `contextPort`.

The effect of this modeling is that the general relationship to `PortPrototype` is already established by `VariableInAtomicSwcInstanceRef` on an abstract level but actually it never makes the generated XML Schema because it is **redefined** by a subclass. In other words, the redefinition replaces the original association as far as the generation algorithm for the XML Schema is concerned.

Class	VariableInAtomicSwcInstanceRef (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components::InstanceRefs			
Note				
Base	ARObject,AtpInstanceRef			
Attribute	Datatype	Mul.	Kind	Note
abstractTargetDataElement	VariableDataPrototype	1	ref	Stereotypes: atpAbstract Tags: xml.sequence Offset=30
base	AtomicSwComponentType	1	ref	Stereotypes: atpDerived Tags: xml.sequence Offset=10
contextPort	PortPrototype	1	ref	Stereotypes: atpAbstract Tags: xml.sequence Offset=20

Table D.2: VariableInAtomicSwcInstanceRef

For clarification, the interpretation of the values of `xml.sequenceOffset` in this particular case is that in the generated XML Schema the attribute `base` comes first, followed by `contextPort`, and then `targetDataElement` concludes the definition of the `InstanceRef` in the XML Schema.

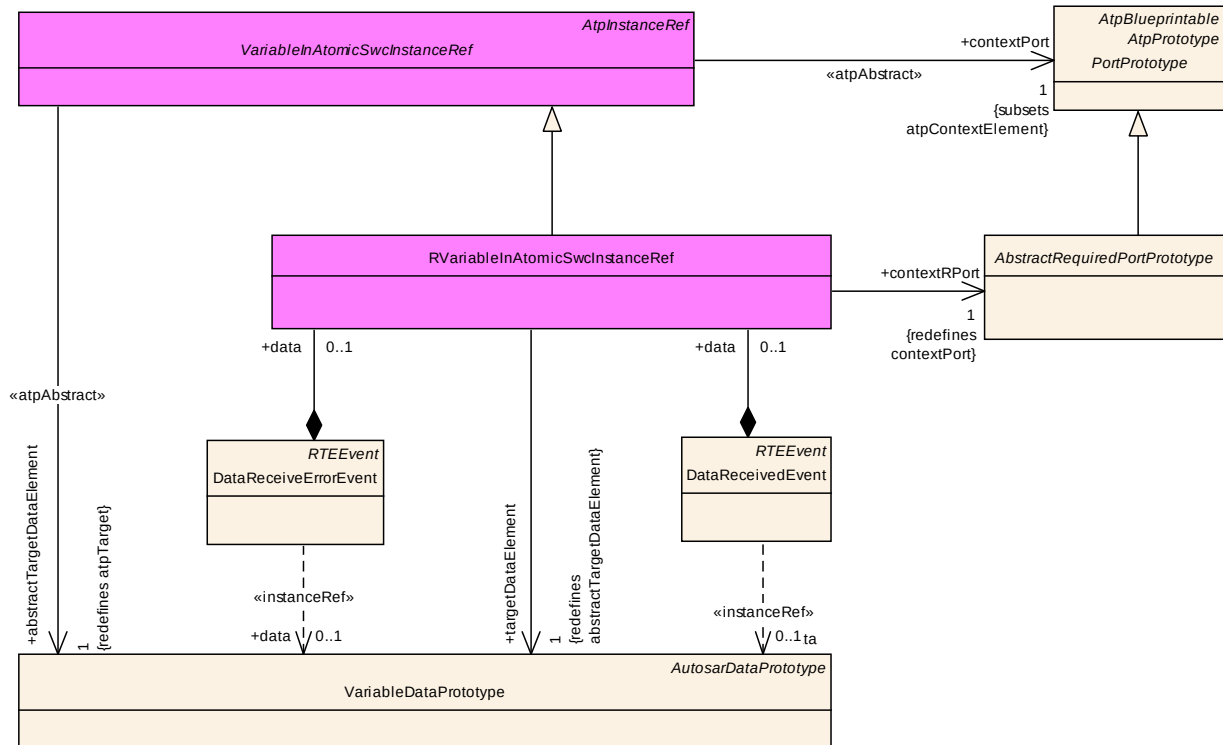


Figure D.2: Concrete modeling of references to *VariableDataPrototype* in the context of an *RPortPrototype*

Class	RVariableInAtomicSwcInstanceRef			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components::InstanceRefs			
Note				
Base	ARObject,AtpInstanceRef,VariableInAtomicSwcInstanceRef			
Attribute	Datatype	Mul.	Kind	Note
contextRPort	AbstractRequiredPortPrototype	1	ref	Tags: xml.sequenceOffset=20
targetDataElement	VariableDataPrototype	1	ref	Tags: xml.sequenceOffset=30

Table D.3: RVariableInAtomicSwcInstanceRef

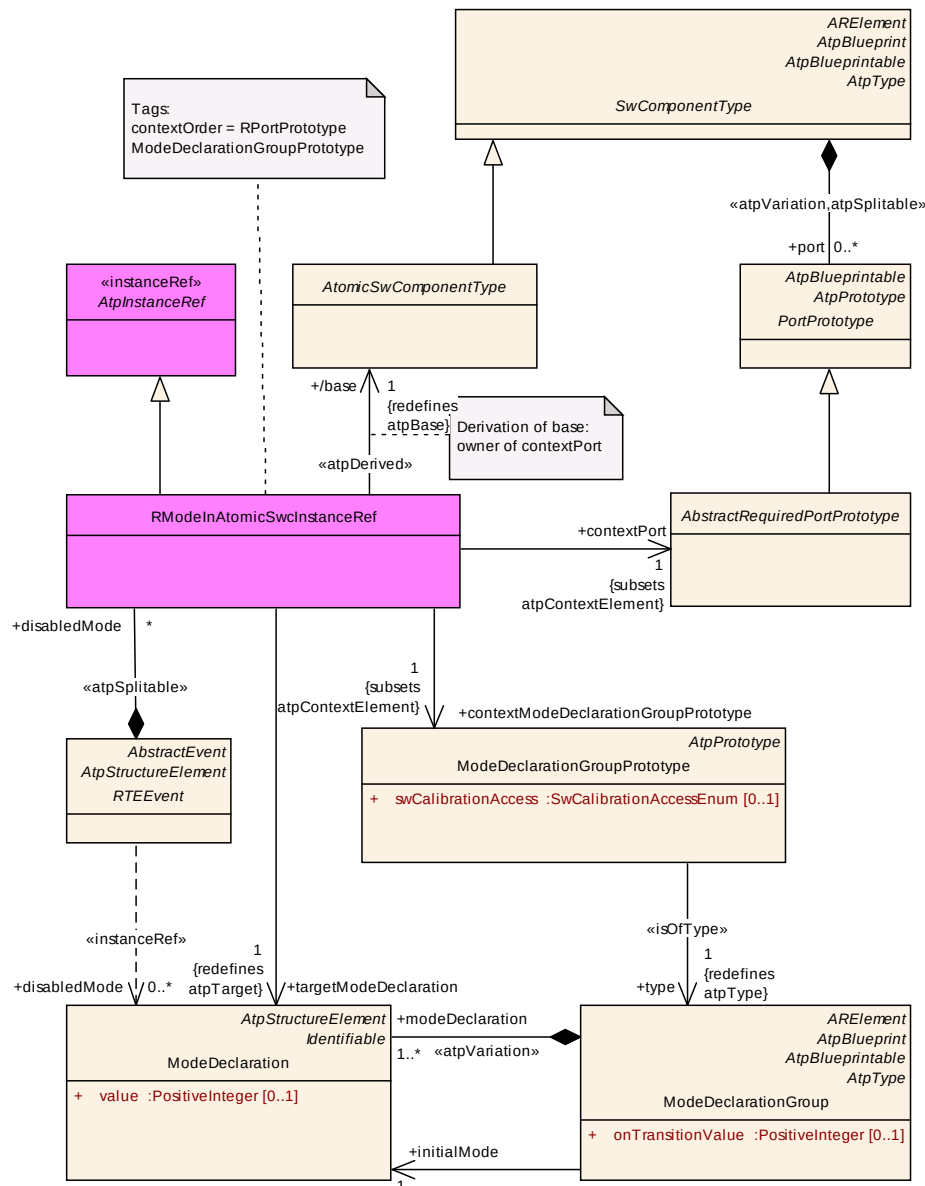


Figure D.3: Modeling of references to `ModeDeclarationGroupPrototype` in the context of an `RPortPrototype`

Class	RModelInAtomicSwcInstanceRef			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components::InstanceRefs			
Note				
Base	ARObject,AtpInstanceRef			
Attribute	Datatype	Mul.	Kind	Note
base	AtomicSwComponentType	1	ref	Stereotypes: atpDerived Tags: xml.sequenceOffset=10
contextModeDeclarationGroupPrototype	ModeDeclarationGroupPrototype	1	ref	Tags: xml.sequenceOffset=30
contextPort	AbstractRequiredPortPrototype	1	ref	Tags: xml.sequenceOffset=20

Attribute	Datatype	Mul.	Kind	Note
targetModeDeclaration	ModeDeclaration	1	ref	Tags: xml.sequenceOffset=40

Table D.4: RModelInAtomicSwcInstanceRef

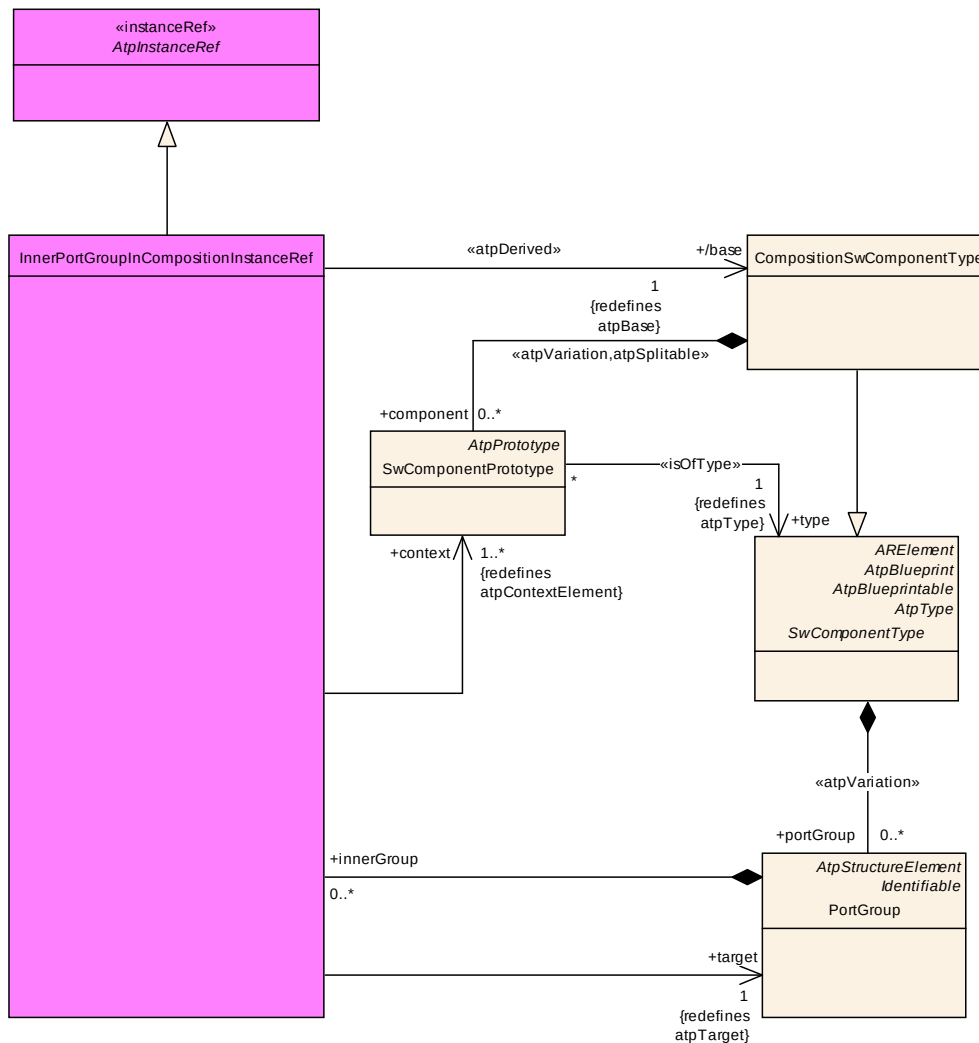


Figure D.4: Modeling of references to PortGroup in the context of a Composition-SwComponentType

Class	InnerPortGroupInCompositionInstanceRef			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components::InstanceRefs			
Note				
Base	ARObject, AtpInstanceRef			
Attribute	Datatype	Mul.	Kind	Note
base	CompositionSwComponentType	1	ref	Stereotypes: atpDerived Tags: xml.sequenceOffset=10
context	SwComponentPrototype	1..*	ref	Tags: xml.sequenceOffset=20

Attribute	Datatype	Mul.	Kind	Note
target	PortGroup	1	ref	Links a PortGroup in a composition to another PortGroup, that is defined in a component which is part of this CompositionSwComponentType. There shall be at most one innerGroup per contained SwComponentPrototype. Tags: xml.sequenceOffset=30

Table D.5: InnerPortGroupInCompositionInstanceRef

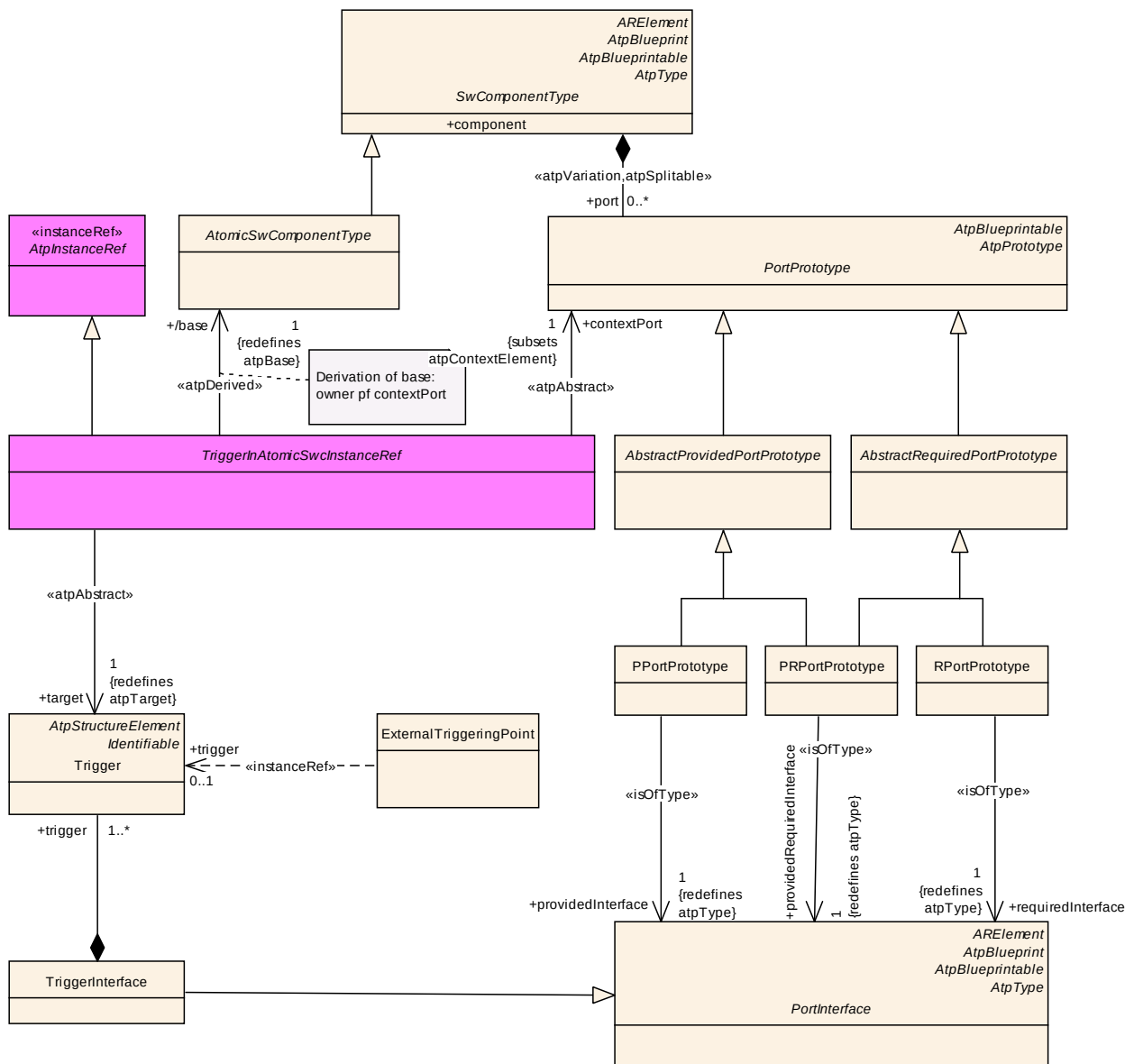


Figure D.5: Abstract modeling of references to `Trigger` in the context of a `AtomicSwComponentType`

Class	TriggerInAtomicSwcInstanceRef (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components::InstanceRefs			
Note				
Base	ARObject, AtpInstanceRef			
Attribute	Datatype	Mul.	Kind	Note
base	AtomicSwComp onentType	1	ref	Stereotypes: atpDerived Tags: xml.sequence Offset=10
contextPort	PortPrototype	1	ref	Stereotypes: atpAbstract Tags: xml.sequence Offset=20
target	Trigger	1	ref	Stereotypes: atpAbstract Tags: xml.sequence Offset=30

Table D.6: TriggerInAtomicSwcInstanceRef

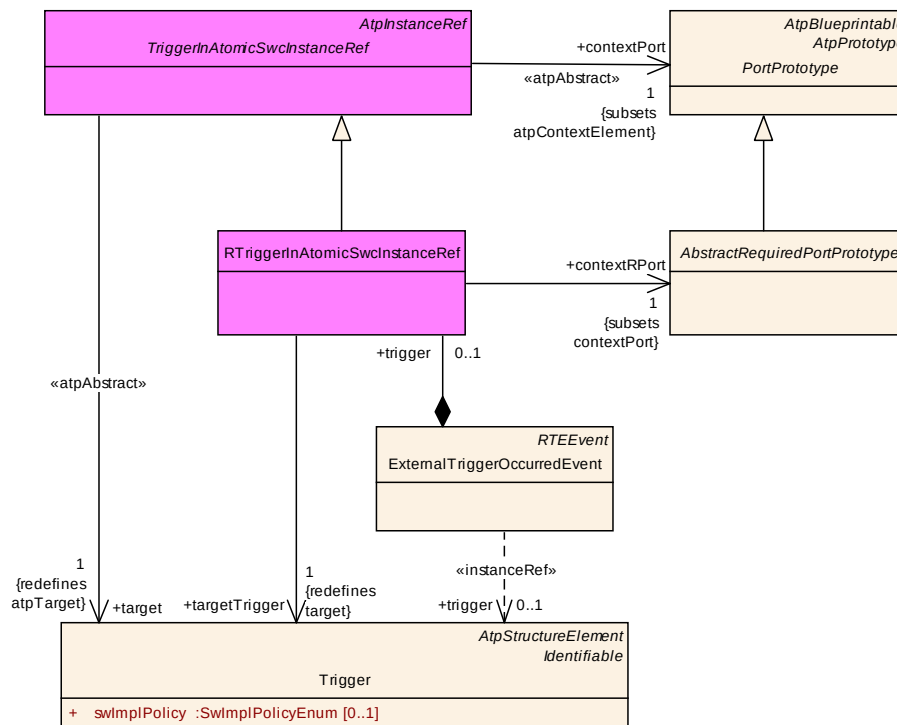


Figure D.6: Concrete modeling of references to **Trigger** in the context of an **RPortPrototype**

Class	RTTriggerInAtomicSwcInstanceRef			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components::InstanceRefs			
Note				
Base	ARObject, AtpInstanceRef, TriggerInAtomicSwcInstanceRef			
Attribute	Datatype	Mul.	Kind	Note
contextRPort	AbstractRequire dPortPrototype	1	ref	Tags: xml.sequenceOffset=20
targetTrigger	Trigger	1	ref	Tags: xml.sequenceOffset=30

Table D.7: RTTriggerInAtomicSwcInstanceRef

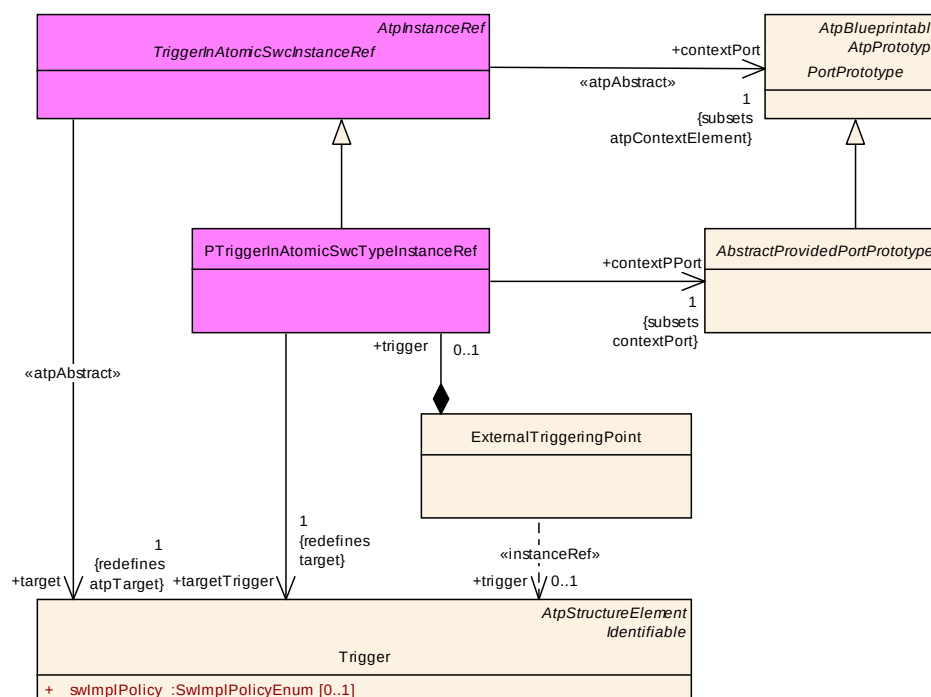


Figure D.7: Concrete modeling of references to `Trigger` in the context of a `PPortPrototype`

Class	PTriggerInAtomicSwcTypeInstanceRef			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components::InstanceRefs			
Note				
Base	ARObject,AtplInstanceRef,TriggerInAtomicSwcInstanceRef			
Attribute	Datatype	Mul.	Kind	Note
contextPPort	AbstractProvidedPortPrototype	1	ref	Tags: xml.sequenceOffset=20
targetTrigger	Trigger	1	ref	Tags: xml.sequenceOffset=30

Table D.8: PTriggerInAtomicSwcTypeInstanceRef

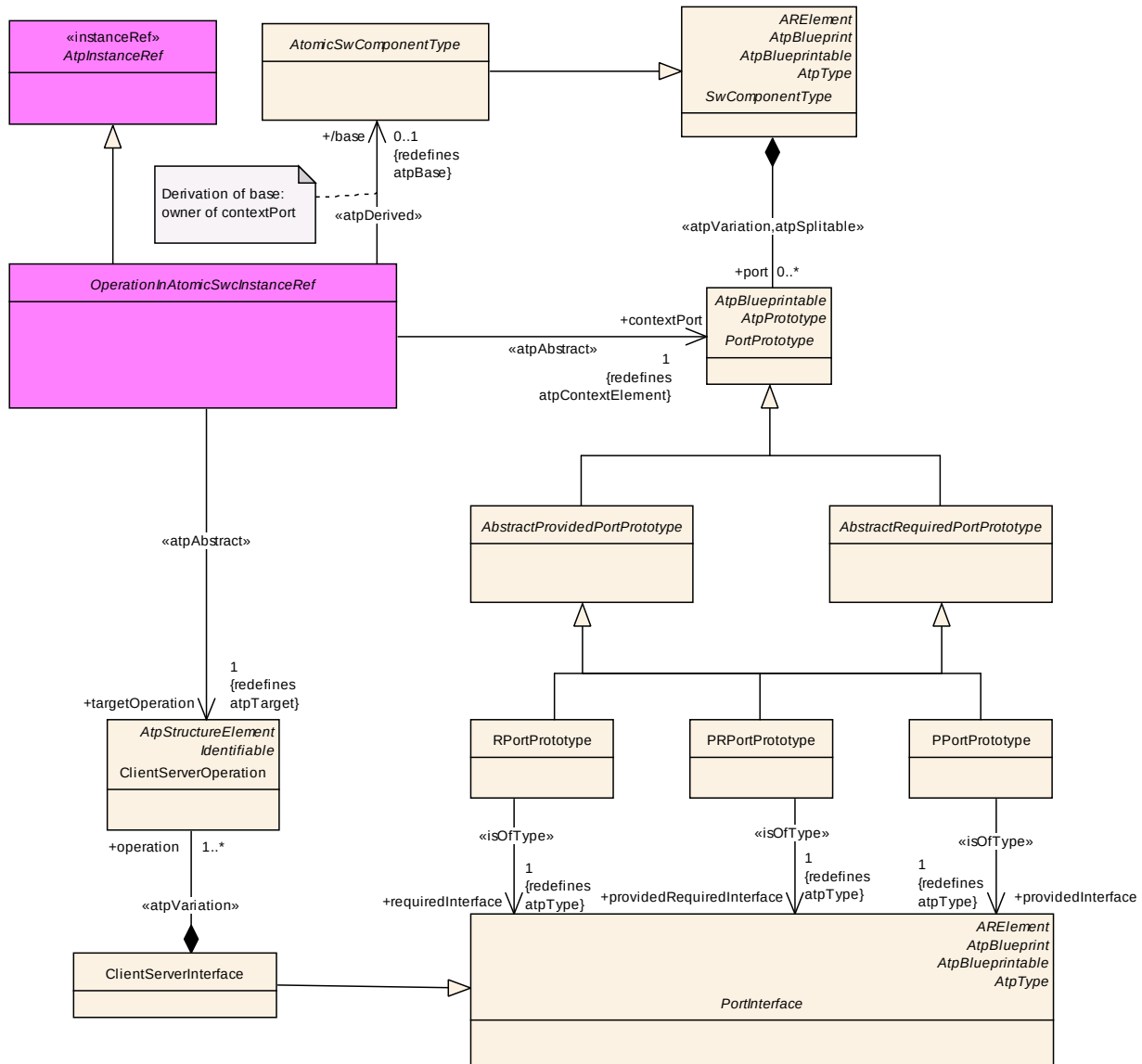


Figure D.8: Abstract modeling of references to *ClientServerOperation* in the context of a *AtomicSwComponentType*

Class	OperationInAtomicSwcInstanceRef (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components::InstanceRefs			
Note				
Base	ARObject, AtpInstanceRef			
Attribute	Datatype	Mul.	Kind	Note
base	AtomicSwComponentType	0..1	ref	Stereotypes: atpDerived Tags: xml.sequence Offset=10
contextPort	PortPrototype	1	ref	Stereotypes: atpAbstract Tags: xml.sequence Offset=20
targetOperation	ClientServerOperation	1	ref	Stereotypes: atpAbstract Tags: xml.sequence Offset=30

Table D.9: OperationInAtomicSwcInstanceRef

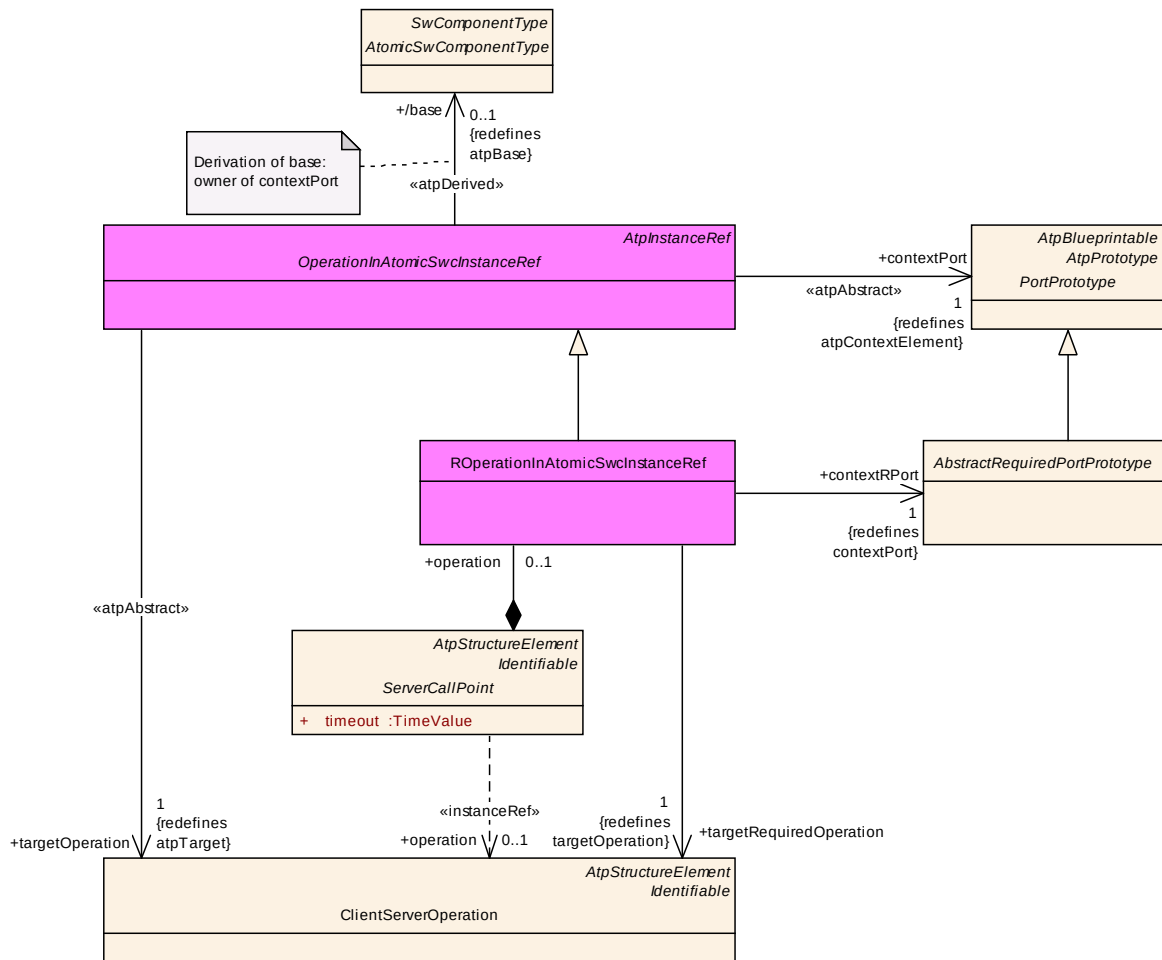


Figure D.9: Concrete modeling of references to *ClientServerOperation* in the context of an *RPortPrototype*

Class	ROperationInAtomicSwcInstanceRef			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components::InstanceRefs			
Note				
Base	ARObject,AtpInstanceRef,OperationInAtomicSwcInstanceRef			
Attribute	Datatype	Mul.	Kind	Note
contextRPort	AbstractRequiredPortPrototype	1	ref	Tags: xml.sequenceOffset=20
targetRequiredOperation	ClientServerOperation	1	ref	Tags: xml.sequenceOffset=30

Table D.10: ROperationInAtomicSwcInstanceRef

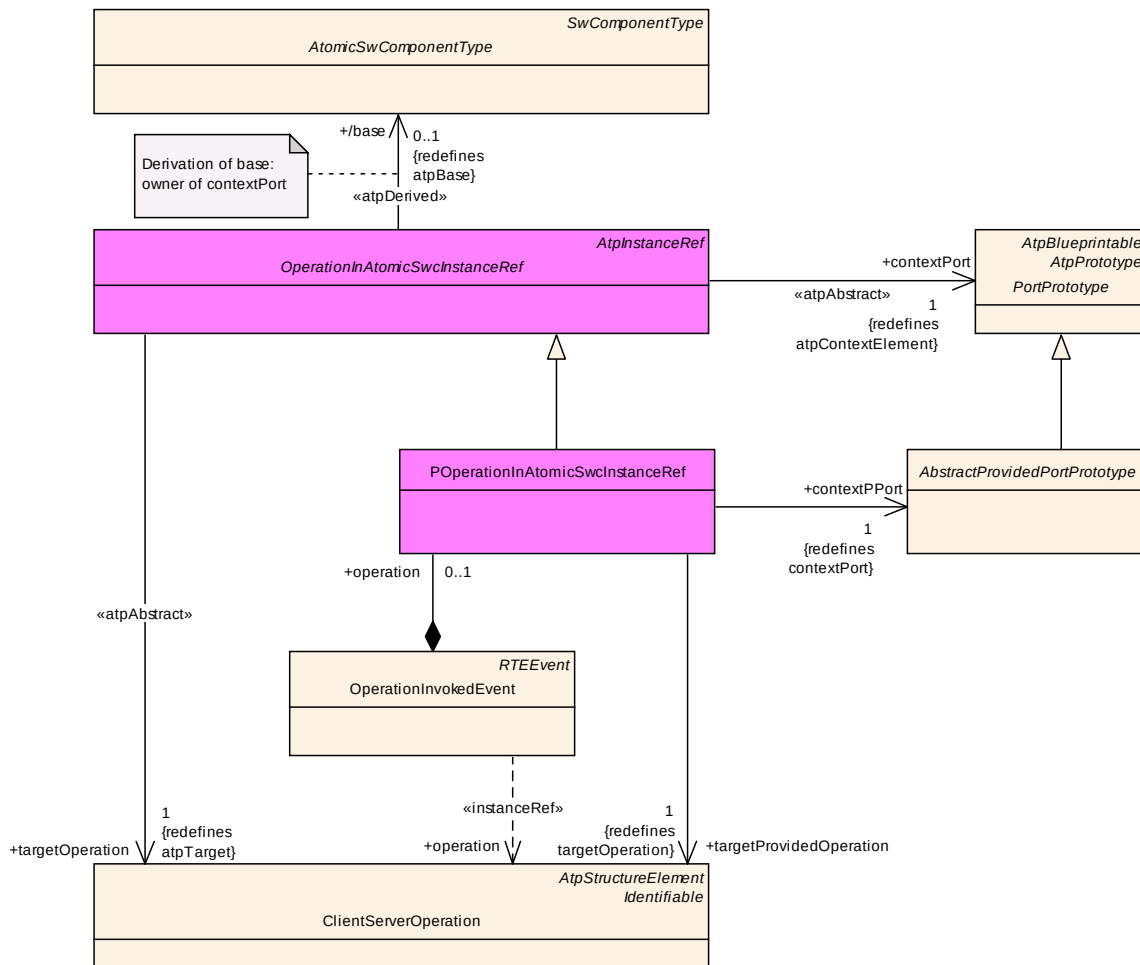


Figure D.10: Concrete modeling of references to *ClientServerOperation* in the context of a *PPortPrototype*

Class	POperationInAtomicSwcInstanceRef			
Package	M2::AUTOSAR Templates::SWComponentTemplate::Components::InstanceRefs			
Note				
Base	ARObject, AtpInstanceRef, OperationInAtomicSwcInstanceRef			
Attribute	Datatype	Mul.	Kind	Note
contextPPort	AbstractProvidedPortPrototype	1	ref	Tags: xml.sequenceOffset=20
targetProvidedOperation	ClientServerOperation	1	ref	Tags: xml.sequenceOffset=30

Table D.11: POperationInAtomicSwcInstanceRef

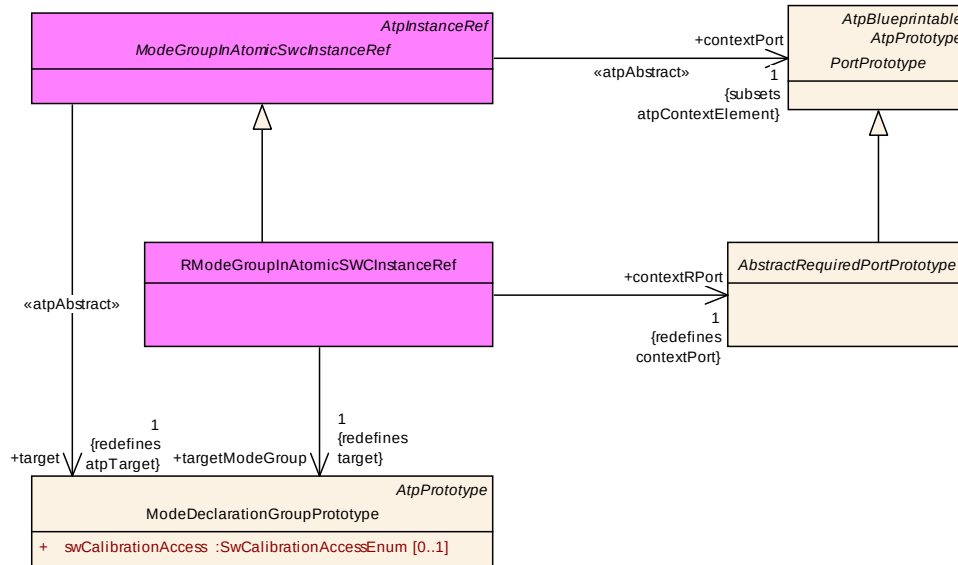


Figure D.11: Concrete modeling of references to `ModeDeclarationGroupPrototype` in the context of an `RPortPrototype`

Class	RModeGroupInAtomicSWCInstanceRef			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components::InstanceRefs			
Note				
Base	ARObject,AtpInstanceRef,ModeGroupInAtomicSwcInstanceRef			
Attribute	Datatype	Mul.	Kind	Note
contextRPort	AbstractRequiredPortPrototype	1	ref	Tags: xml.sequenceOffset=20
targetModeGroup	ModeDeclarationGroupPrototype	1	ref	Tags: xml.sequenceOffset=30

Table D.12: RModeGroupInAtomicSWCInstanceRef

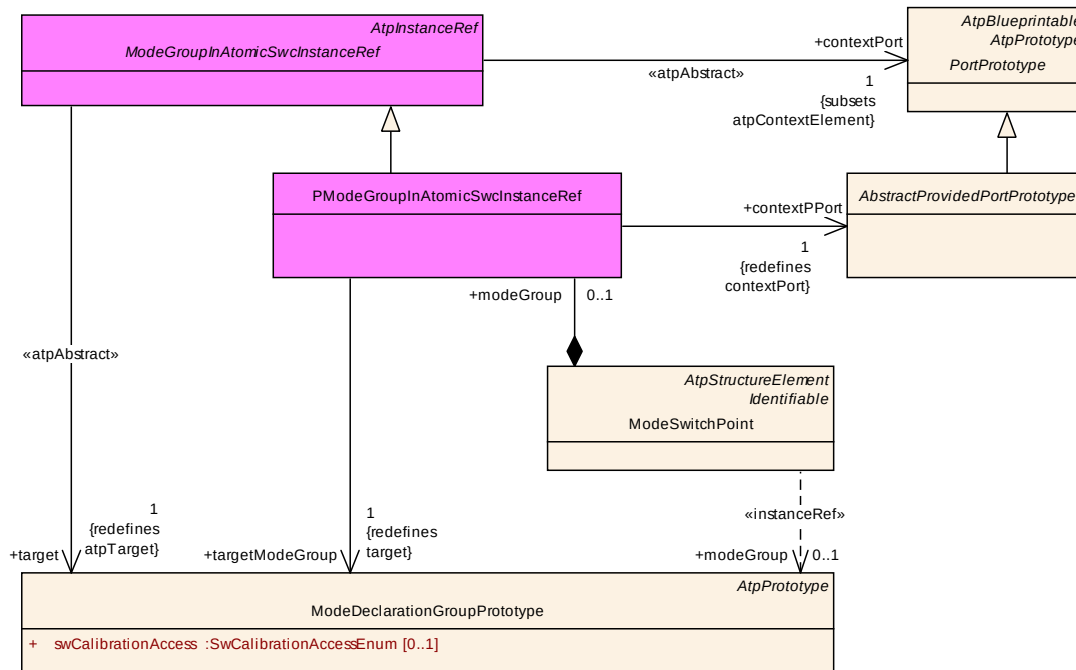


Figure D.12: Concrete modeling of references to `ModeDeclarationGroupPrototype` in the context of a `PPortPrototype`

Class	PModeGroupInAtomicSwcInstanceRef			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components::InstanceRefs			
Note				
Base	ARObject, AtpInstanceRef, ModeGroupInAtomicSwcInstanceRef			
Attribute	Datatype	Mul.	Kind	Note
contextPPort	AbstractProvidedPortPrototype	1	ref	Tags: xml.sequenceOffset=20
targetModeGroup	ModeDeclarationGroupPrototype	1	ref	Tags: xml.sequenceOffset=30

Table D.13: PModeGroupInAtomicSwcInstanceRef

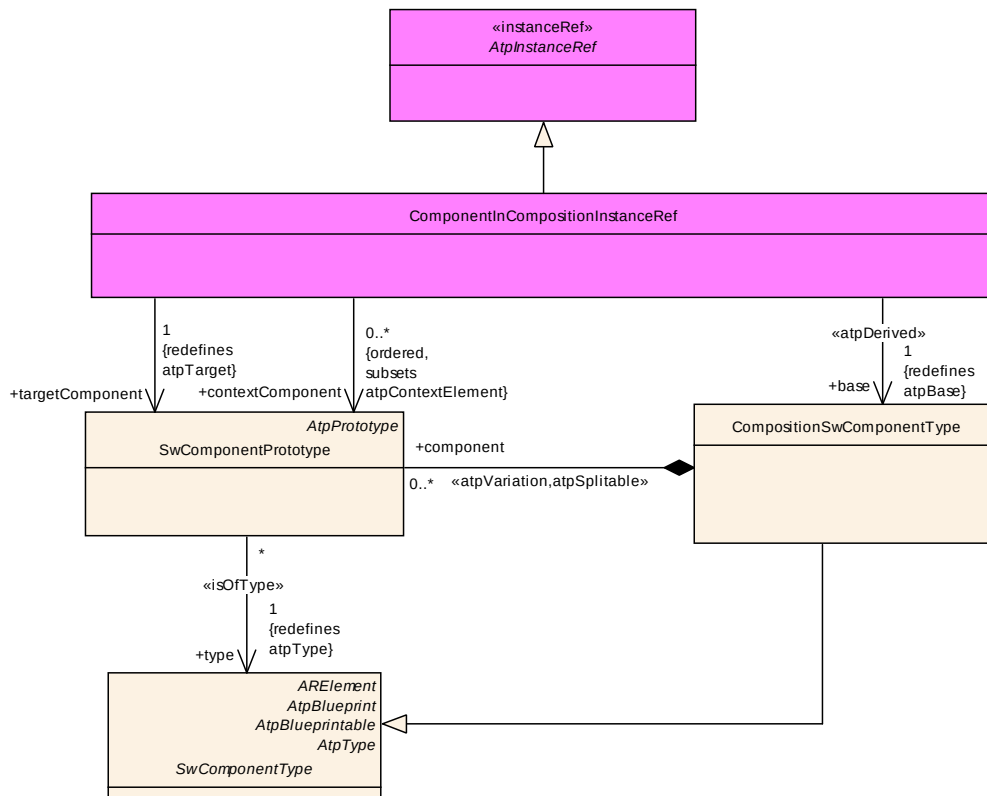


Figure D.13: Concrete modeling of references to **SwComponentPrototype in the context of a **CompositionSwComponentType****

Class	ComponentInCompositionInstanceRef				
Package	M2::AUTOSARTemplates::SWComponentTemplate::Composition::InstanceRefs				
Note	The ComponentInCompositionInstanceRef points to a concrete SwComponentPrototype within a CompositionSwComponentType.				
Base	ARObject,AtpInstanceRef				
Attribute	Datatype	Mul.	Kind	Note	
base	CompositionSwComponentType	1	ref	Stereotypes: atpDerived Tags: xml.sequenceOffset=10	
contextComponent (ordered)	SwComponentPrototype	*	ref	Tags: xml.sequenceOffset=20	
targetComponent	SwComponentPrototype	1	ref	Tags: xml.sequenceOffset=30	

Table D.14: ComponentInCompositionInstanceRef

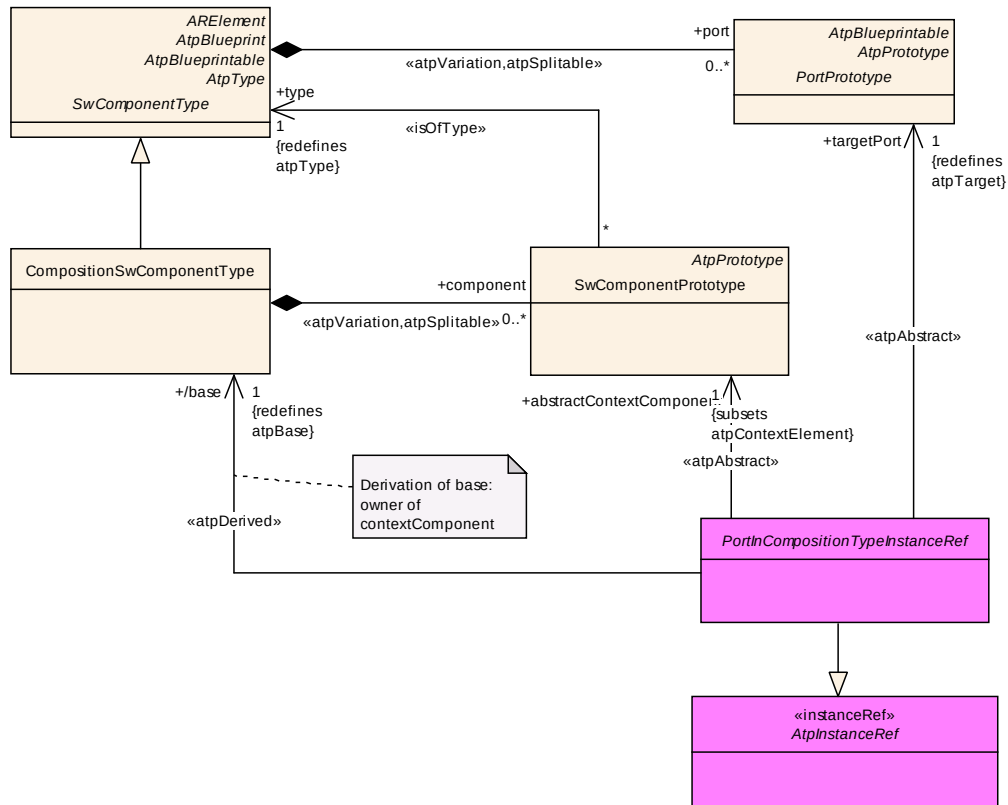


Figure D.14: Abstract modeling of references to **PortPrototype** in the context of a **CompositionSwComponentType**

Class	PortInCompositionTypeInstanceRef (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Composition::InstanceRefs			
Note				
Base	ARObject,AtpInstanceRef			
Attribute	Datatype	Mul.	Kind	Note
abstractContextComponent	SwComponentPrototype	1	ref	Stereotypes: atpAbstract Tags: xml.sequence Offset=20
base	CompositionSwComponentType	1	ref	Stereotypes: atpDerived Tags: xml.sequence Offset=10
targetPort	PortPrototype	1	ref	Stereotypes: atpAbstract Tags: xml.sequence Offset=30

Table D.15: PortInCompositionTypeInstanceRef

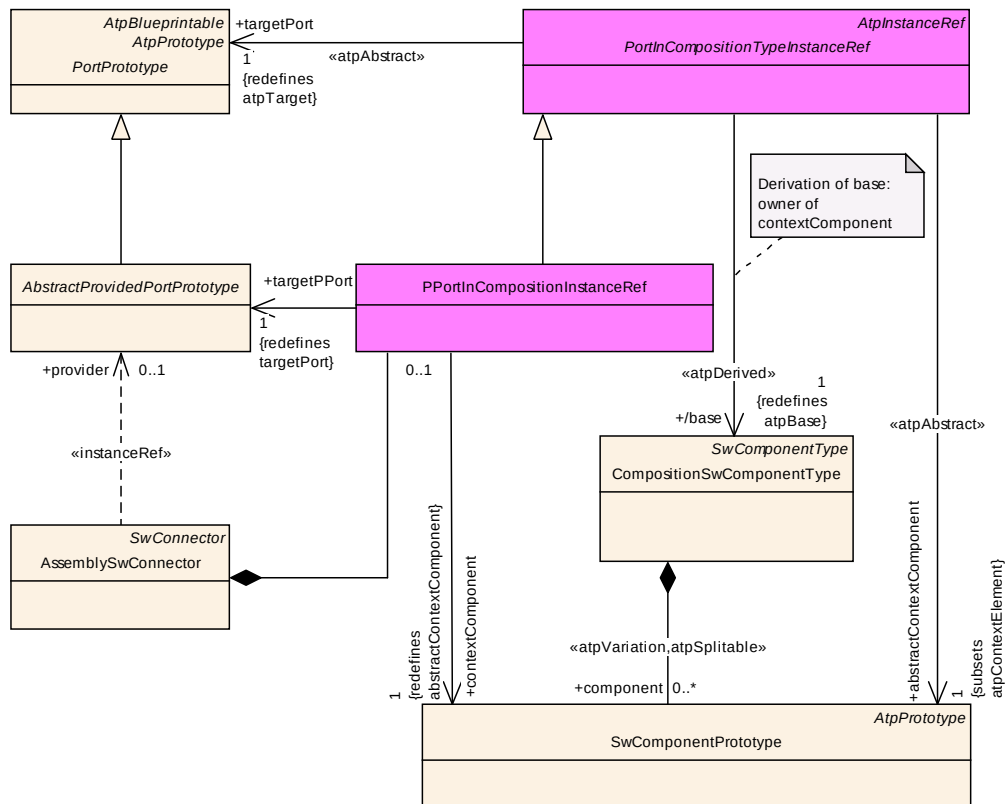


Figure D.15: Concrete modeling of references to **PPortInCompositionInstanceRef** in the context of a **CompositionSwComponentType**

Class	PPortInCompositionInstanceRef			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Composition::InstanceRefs			
Note				
Base	ARObject, AtpInstanceRef, PortInCompositionTypeInstanceRef			
Attribute	Datatype	Mul.	Kind	Note
contextComponent	SwComponentPrototype	1	ref	Tags: xml.sequenceOffset=20
targetPPort	AbstractProvidedPortPrototype	1	ref	Tags: xml.sequenceOffset=30

Table D.16: PPortInCompositionInstanceRef

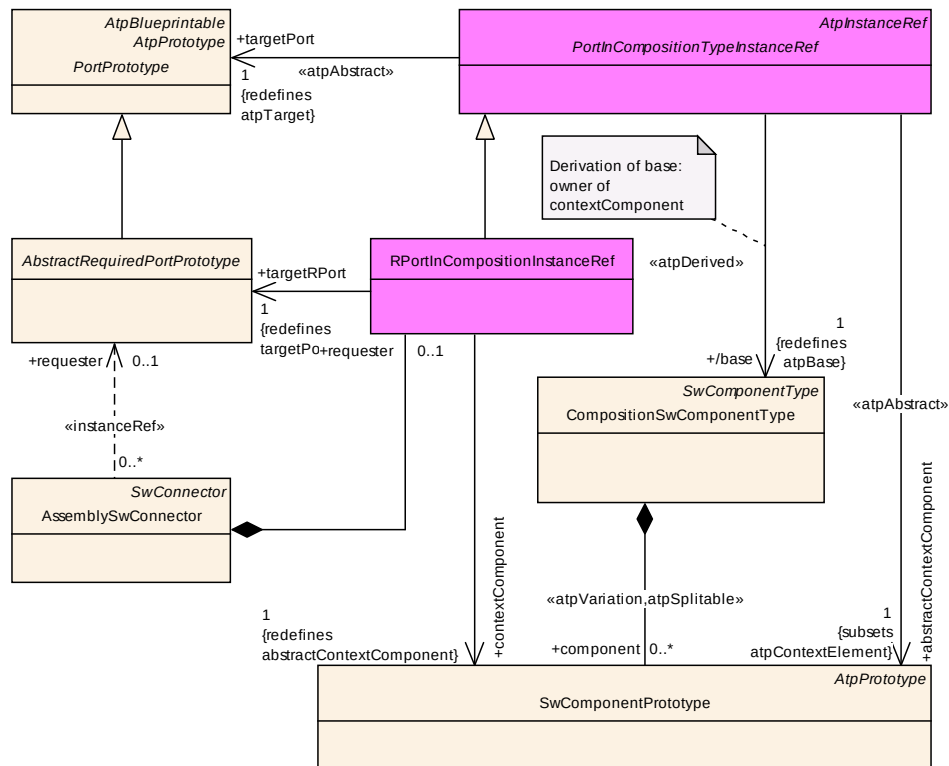


Figure D.16: Concrete modeling of references to **RPortPrototype** in the context of a **CompositionSwComponentType**

Class	RPortInCompositionInstanceRef			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Composition::InstanceRefs			
Note				
Base	ARObject, AtpInstanceRef, PortInCompositionTypeInstanceRef			
Attribute	Datatype	Mul.	Kind	Note
contextComponent	SwComponentPrototype	1	ref	Tags: xml.sequenceOffset=20
targetRPort	AbstractRequiredPortPrototype	1	ref	Tags: xml.sequenceOffset=30

Table D.17: RPortInCompositionInstanceRef

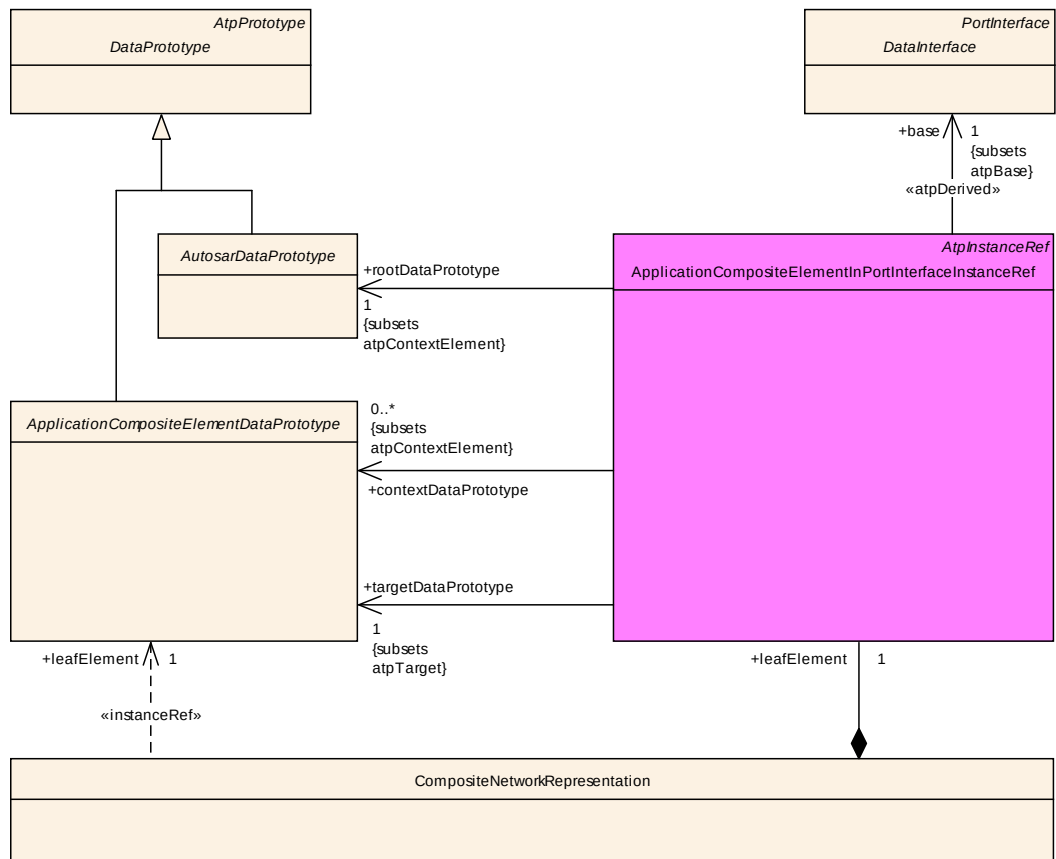


Figure D.17: Modeling of references to **ApplicationCompositeElementDataPrototype for the purpose of defining a network representation**

D.2.2 Definition of implicit Communication Behavior

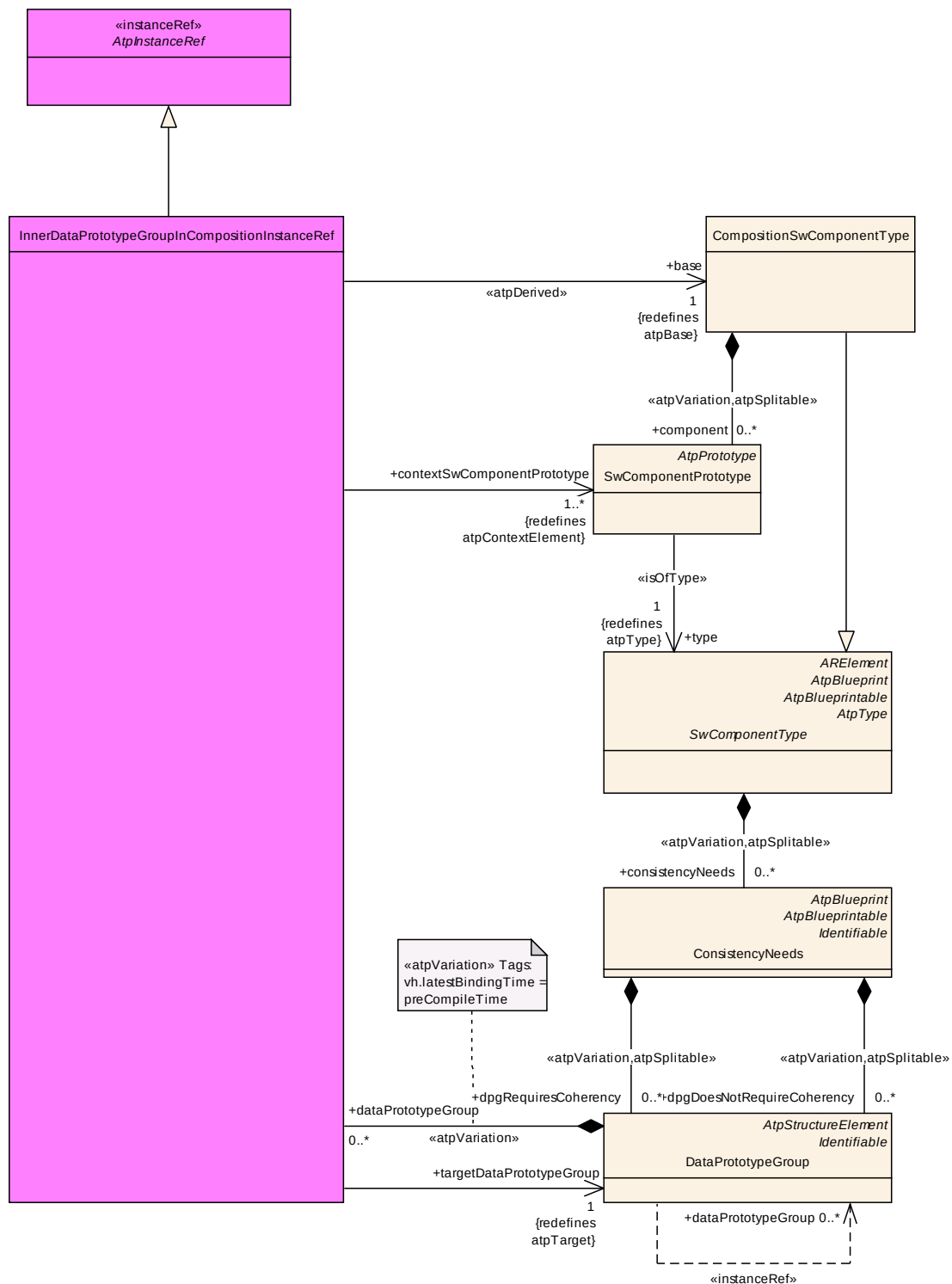


Figure D.18: Modeling of references to `DataPrototypeGroup` in the context of a `CompositionSwComponentType`

Class	InnerDataPrototypeGroupInCompositionInstanceRef			
Package	M2::AUTOSARTemplates::SWComponentTemplate::ImplicitCommunicationBehavior::InstanceRef			
Note	This meta-class represents the ability to define an InstanceRef to a nested DataPrototypeGroup			
Base	ARObject,AtpInstanceRef			
Attribute	Datatype	Mul.	Kind	Note
base	CompositionSwComponentType	1	ref	This represents the base of the instanceRef. Stereotypes: atpDerived Tags: xml.sequenceOffset=10
contextSwComponentPrototype	SwComponentPrototype	1..*	ref	This represents the nested structure of SwComponentPrototypes. Tags: xml.sequenceOffset=20
targetDataPrototypeGroup	DataPrototypeGroup	1	ref	This represents the target of the InstanceRef Tags: xml.sequenceOffset=30

Table D.18: InnerDataPrototypeGroupInCompositionInstanceRef

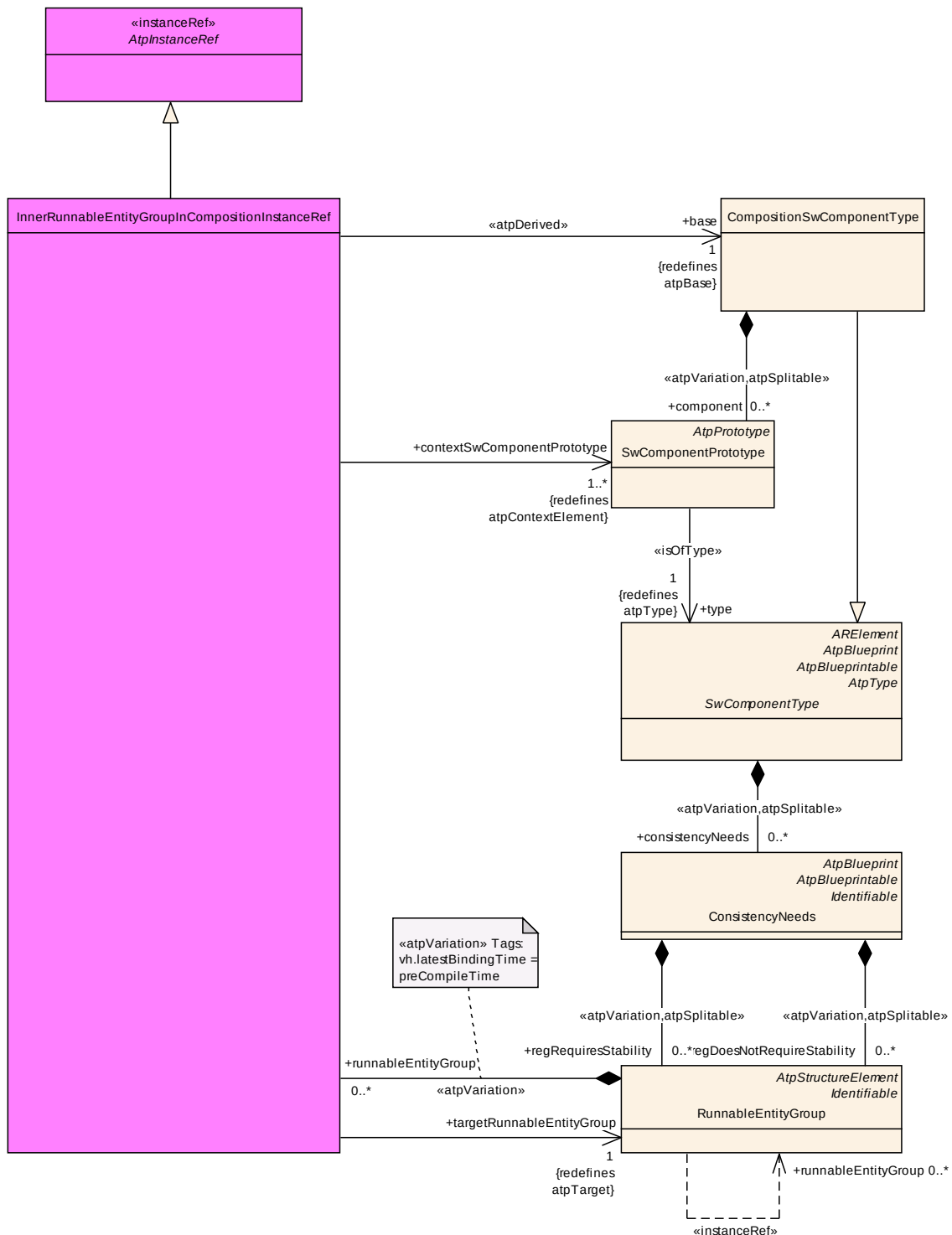


Figure D.19: Modeling of references to RunnableEntityGroup in the context of a CompositionSwComponentType

Class	InnerRunnableEntityGroupInCompositionInstanceRef			
Package	M2::AUTOSARTemplates::SWComponentTemplate::ImplicitCommunicationBehavior::InstanceRef			
Note	This meta-class represents the ability to define an InstanceRef to a nested RunnableEntityGroup.			
Base	ARObject,AtpInstanceRef			
Attribute	Datatype	Mul.	Kind	Note
base	CompositionSwComponentType	1	ref	This represents the base of the InstanceRef. Stereotypes: atpDerived Tags: xml.sequenceOffset=10
contextSwComponentPrototype	SwComponentPrototype	1..*	ref	This represents the nested structure of SwComponentPrototypes. Tags: xml.sequenceOffset=20
targetRunnableEntityGroup	RunnableEntityGroup	1	ref	This represents the target association of the InstanceRef. Tags: xml.sequenceOffset=30

Table D.19: InnerRunnableEntityGroupInCompositionInstanceRef

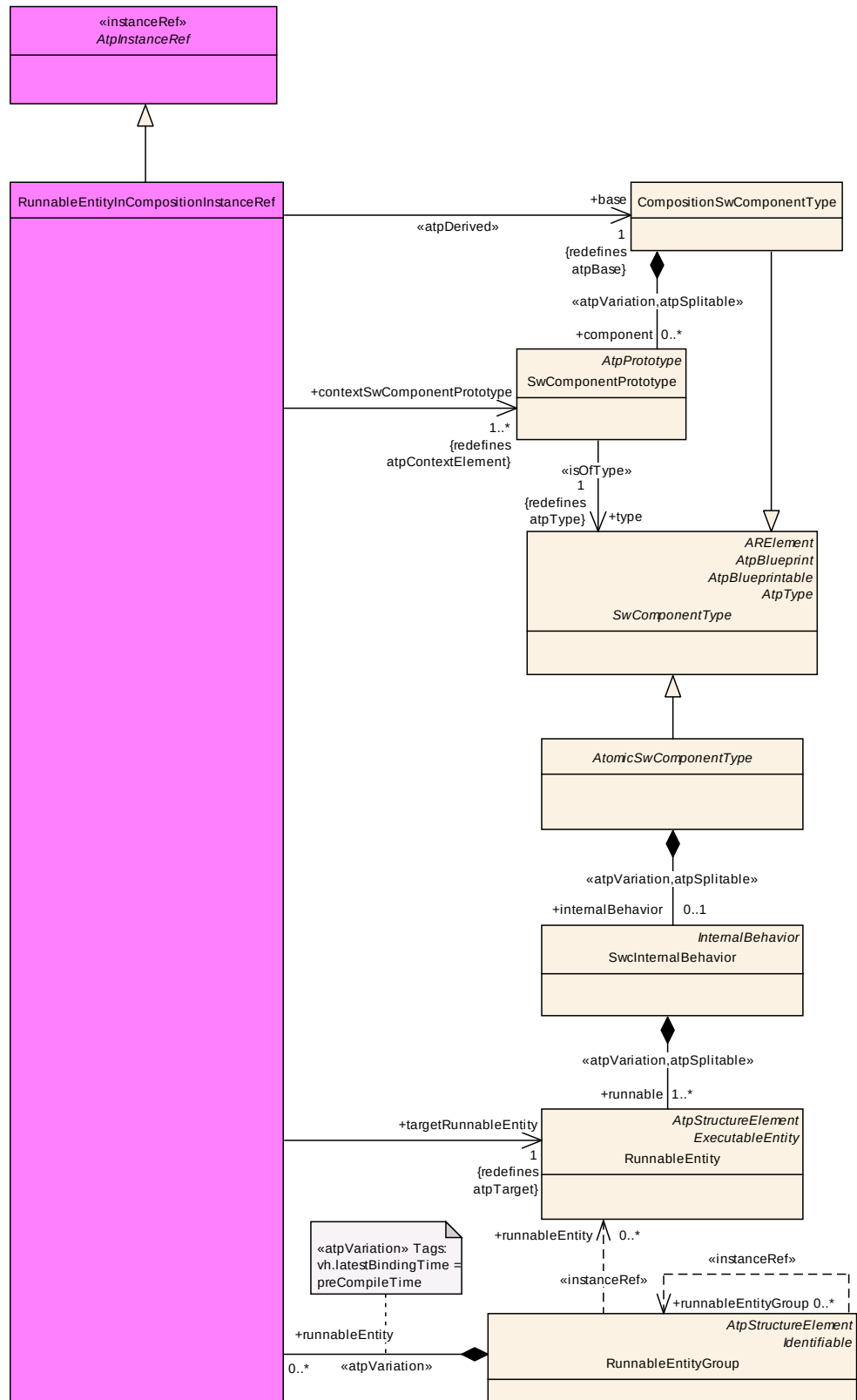


Figure D.20: Modeling of references to **RunnableEntity** in the context of a **CompositionSwComponentType** from the point of view of a **RunnableEntityGroup**

Class	RunnableEntityInCompositionInstanceRef			
Package	M2::AUTOSARTemplates::SWComponentTemplate::ImplicitCommunicationBehavior::InstanceRef			
Note	This meta-class represents the ability to define an InstanceRef to a RunnableEntity in the context of a CompositionSwComponentType.			
Base	ARObject,AtpInstanceRef			
Attribute	Datatype	Mul.	Kind	Note
base	CompositionSwComponentType	1	ref	This represents the base of the InstanceRef. Stereotypes: atpDerived Tags: xml.sequenceOffset=10
contextSwComponentPrototype	SwComponentPrototype	1..*	ref	This represents the nested structure of SwComponentPrototypes. Tags: xml.sequenceOffset=20
targetRunnableEntity	RunnableEntity	1	ref	This represents the target RunnableEntity. Tags: xml.sequenceOffset=30

Table D.20: RunnableEntityInCompositionInstanceRef

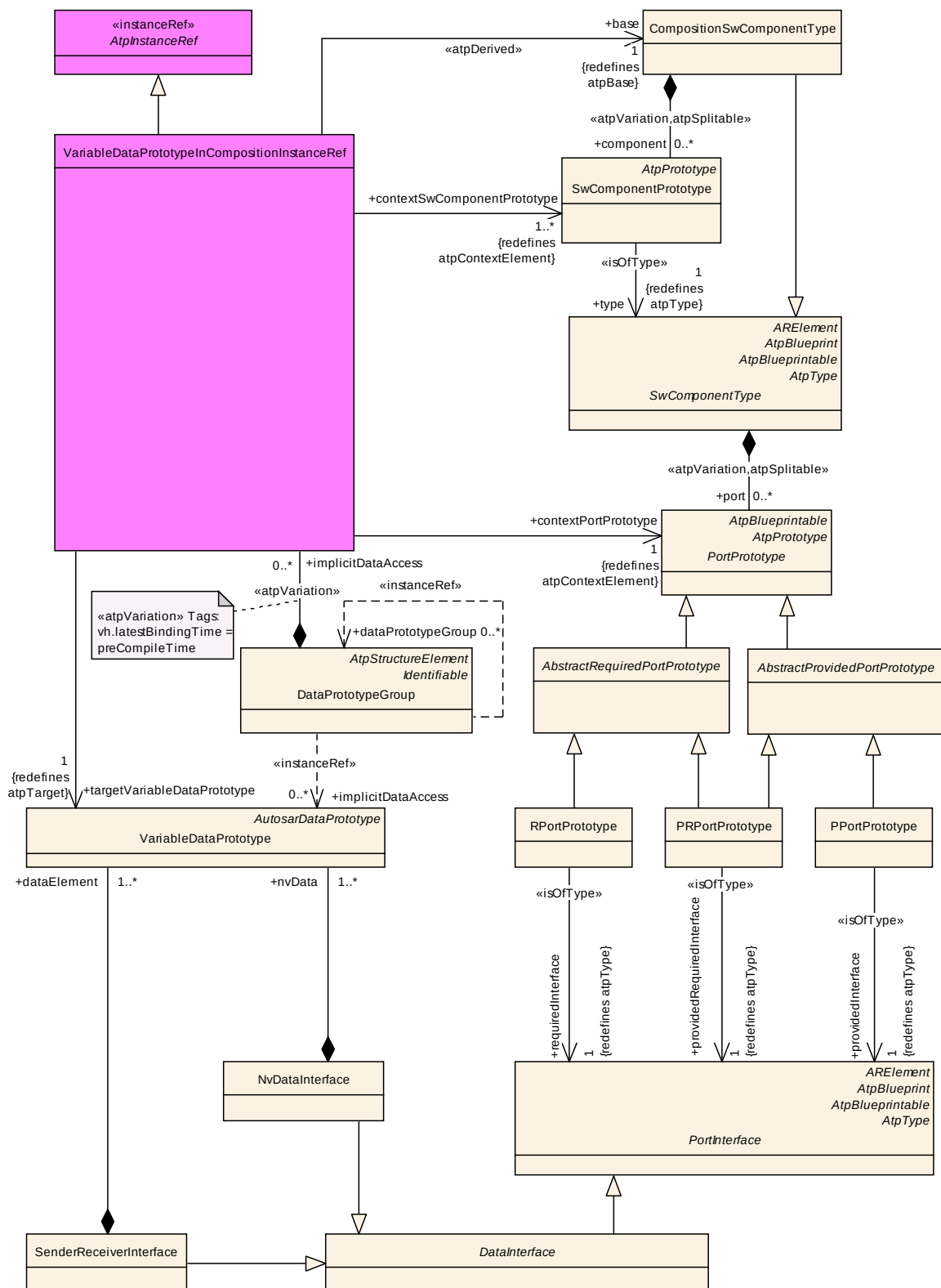


Figure D.21: Modeling of references to `VariableDataPrototype` in the context of a `CompositionSwComponentType` from the point of view of a `RunnableEntityGroup`

Class	VariableDataPrototypeInCompositionInstanceRef			
Package	M2::AUTOSARTemplates::SWComponentTemplate::ImplicitCommunicationBehavior::InstanceRef			
Note	This meta-class represents the ability to define an InstanceRef to a VariableDataPrototype in the context of a CompositionSwComponentType.			
Base	ARObject,AtpInstanceRef			
Attribute	Datatype	Mul.	Kind	Note
base	CompositionSwComponentType	1	ref	This represents the base of the InstanceRef. Stereotypes: atpDerived Tags: xml.sequenceOffset=10
contextPortPrototype	PortPrototype	1	ref	This represents a reference to a context PortPrototype. Tags: xml.sequenceOffset=30
contextSwComponentPrototype	SwComponentPrototype	1..*	ref	This represents the nested structure of SwComponentPrototypes. Tags: xml.sequenceOffset=20
targetVariableDataPrototype	VariableDataPrototype	1	ref	This represents the target VariableDataPrototype. Tags: xml.sequenceOffset=40

Table D.21: VariableDataPrototypeInCompositionInstanceRef

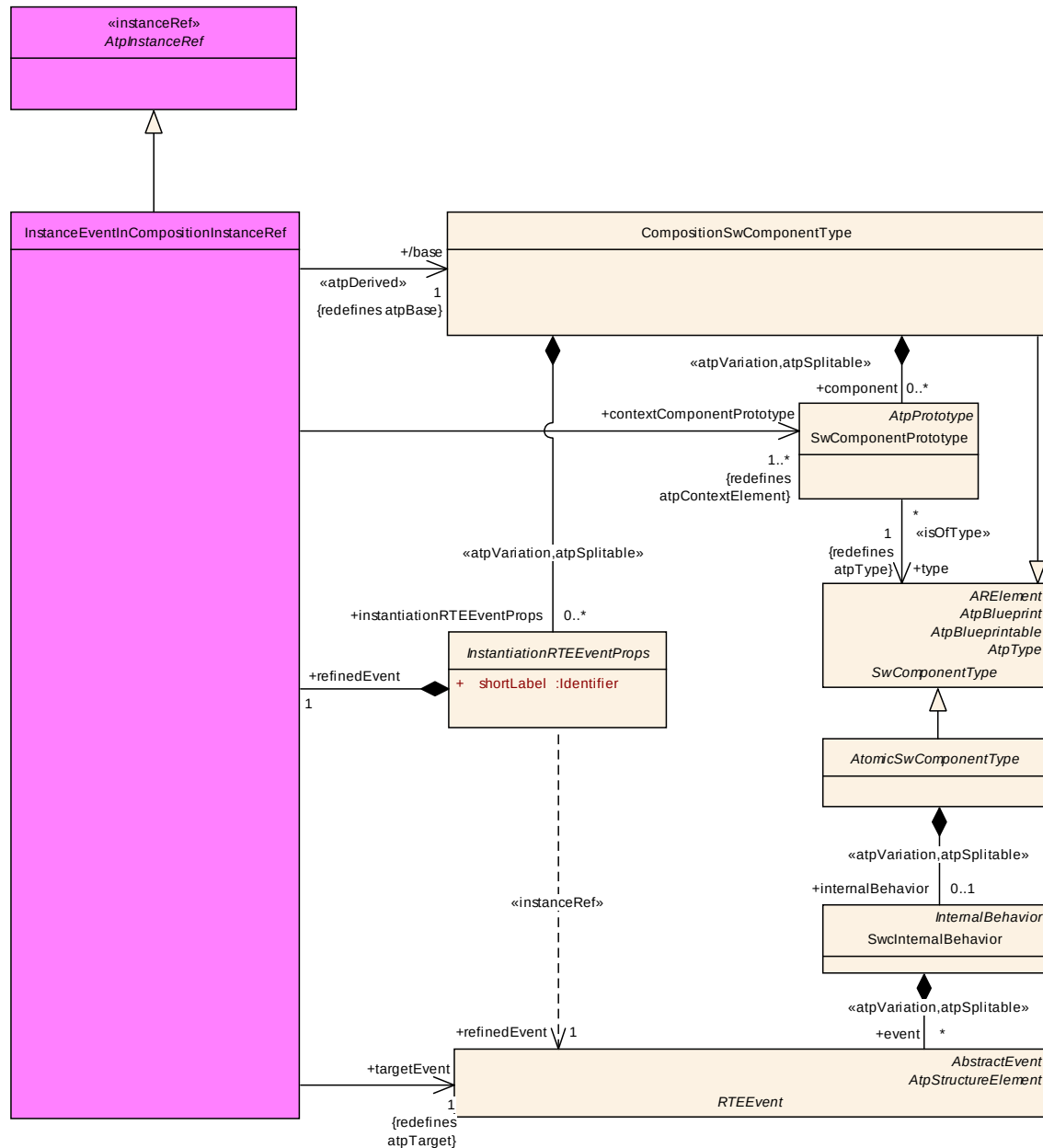


Figure D.22: Modeling of references to **RTEEvent** in the context of a **InstantiationRTEEventProps** from the point of view of a **CompositionSwComponentType**

Class	InstanceEventInCompositionInstanceRef			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Composition::InstanceRefs			
Note				
Base	ARObject,AtpInstanceRef			
Attribute	Datatype	Mul.	Kind	Note
base	CompositionSwComponentType	1	ref	Stereotypes: atpDerived Tags: xml.sequenceOffset=10
contextComponentPrototype	SwComponentPrototype	1..*	ref	Tags: xml.sequenceOffset=20

<i>Attribute</i>	<i>Datatype</i>	<i>Mul.</i>	<i>Kind</i>	<i>Note</i>
targetEvent	RTEEvent	1	ref	Tags: xml.sequenceOffset=30

Table D.22: InstanceEventInCompositionInstanceRef

D.2.3 Internal Behavior

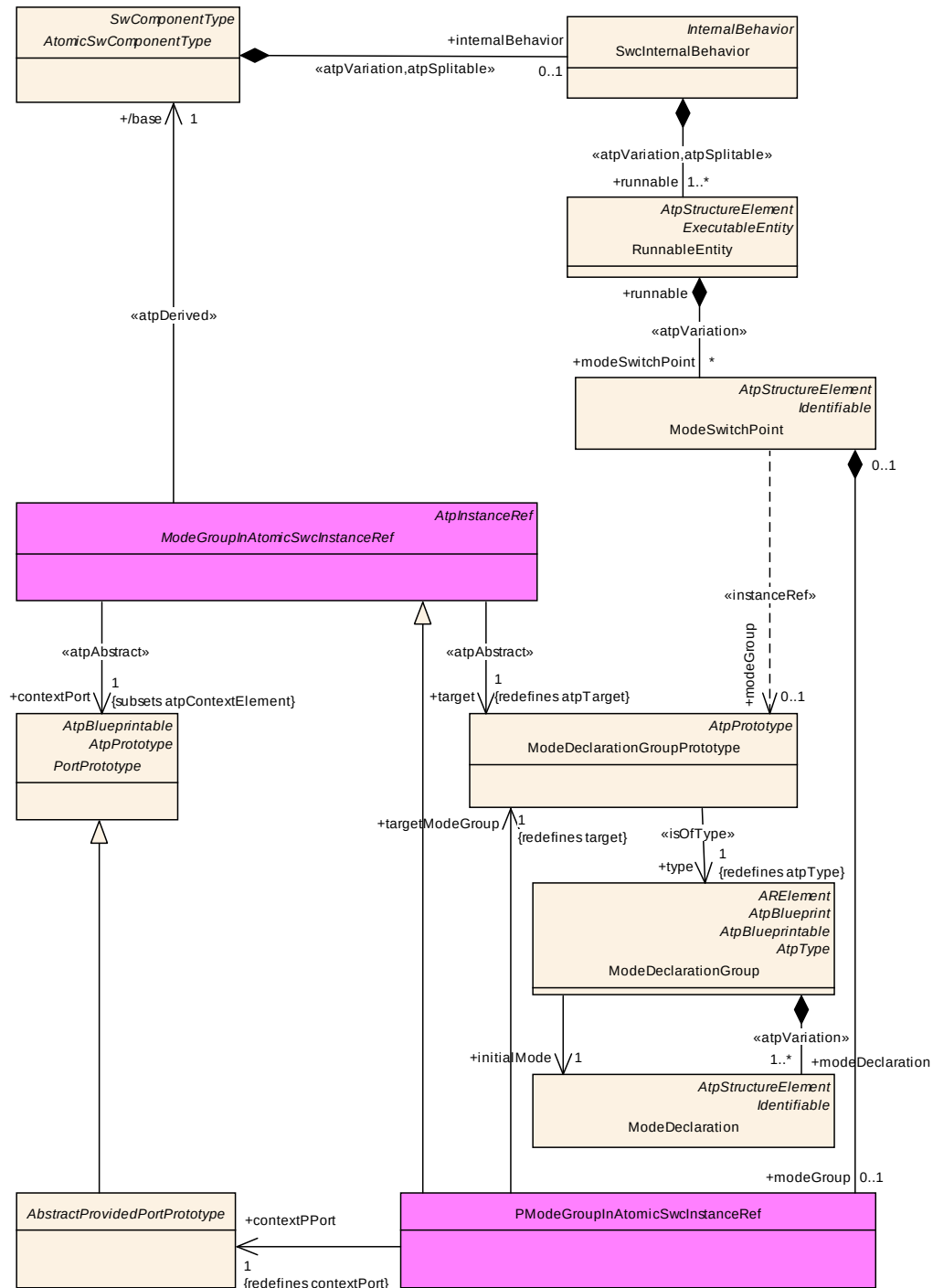


Figure D.23: Modeling of references to provided **ModeDeclarationGroupPrototype in the context of an **AtomicSwComponentType****

Class	ModeGroupInAtomicSwcInstanceRef (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components::InstanceRefs			
Note				
Base	ARObject, AtpInstanceRef			
Attribute	Datatype	Mul.	Kind	Note
base	AtomicSwComponentType	1	ref	Stereotypes: atpDerived Tags: xml.sequence Offset=10
contextPort	PortPrototype	1	ref	Stereotypes: atpAbstract Tags: xml.sequence Offset=20
target	ModeDeclarationGroupPrototype	1	ref	Stereotypes: atpAbstract Tags: xml.sequence Offset=30

Table D.23: ModeGroupInAtomicSwcInstanceRef

Class	PModeGroupInAtomicSwcInstanceRef			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components::InstanceRefs			
Note				
Base	ARObject, AtpInstanceRef, ModeGroupInAtomicSwcInstanceRef			
Attribute	Datatype	Mul.	Kind	Note
contextPPort	AbstractProvidedPortPrototype	1	ref	Tags: xml.sequenceOffset=20
targetModeGroup	ModeDeclarationGroupPrototype	1	ref	Tags: xml.sequenceOffset=30

Table D.24: PModeGroupInAtomicSwcInstanceRef

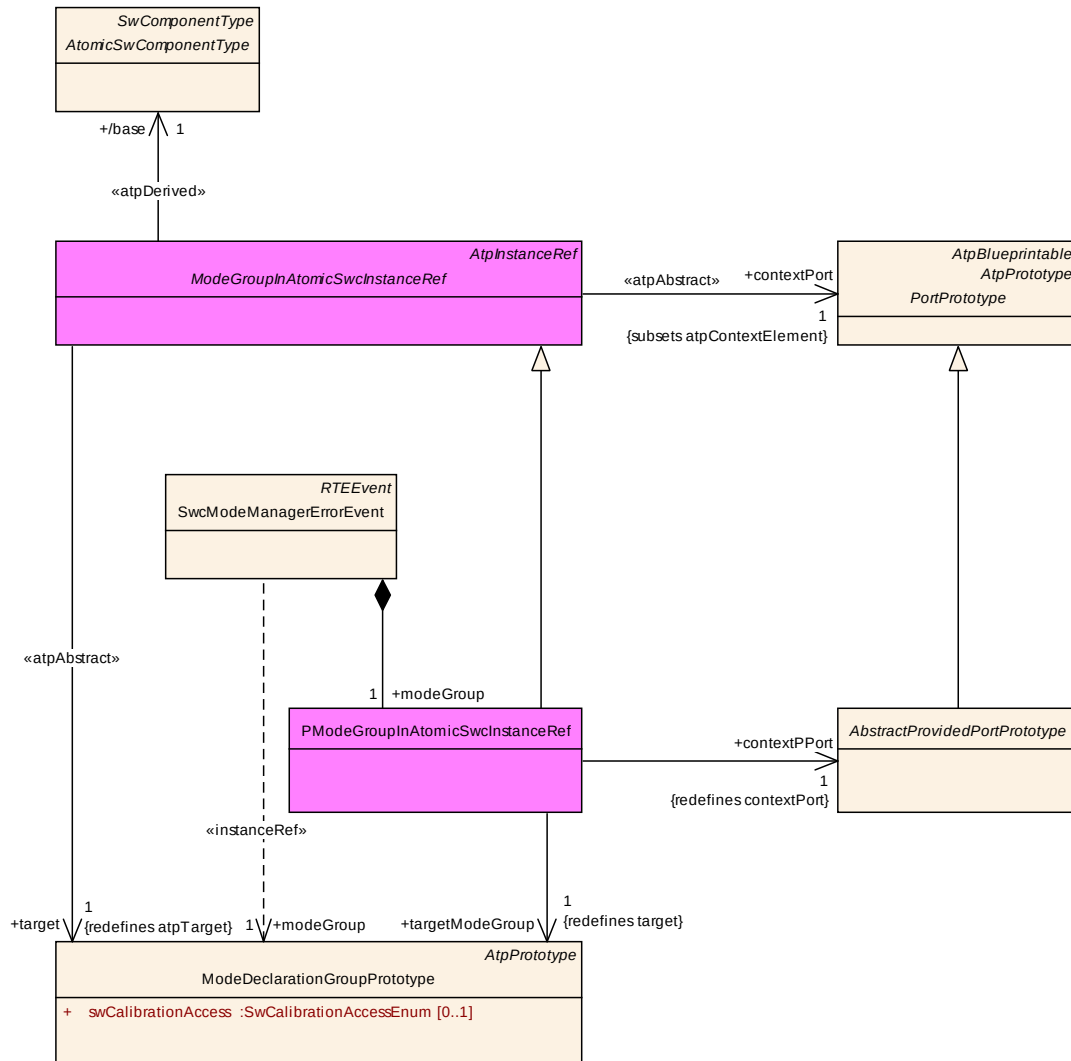


Figure D.24: Modeling of references to provided `ModeDeclarationGroupPrototype` to be used by `SwcModeManagerErrorEvent`

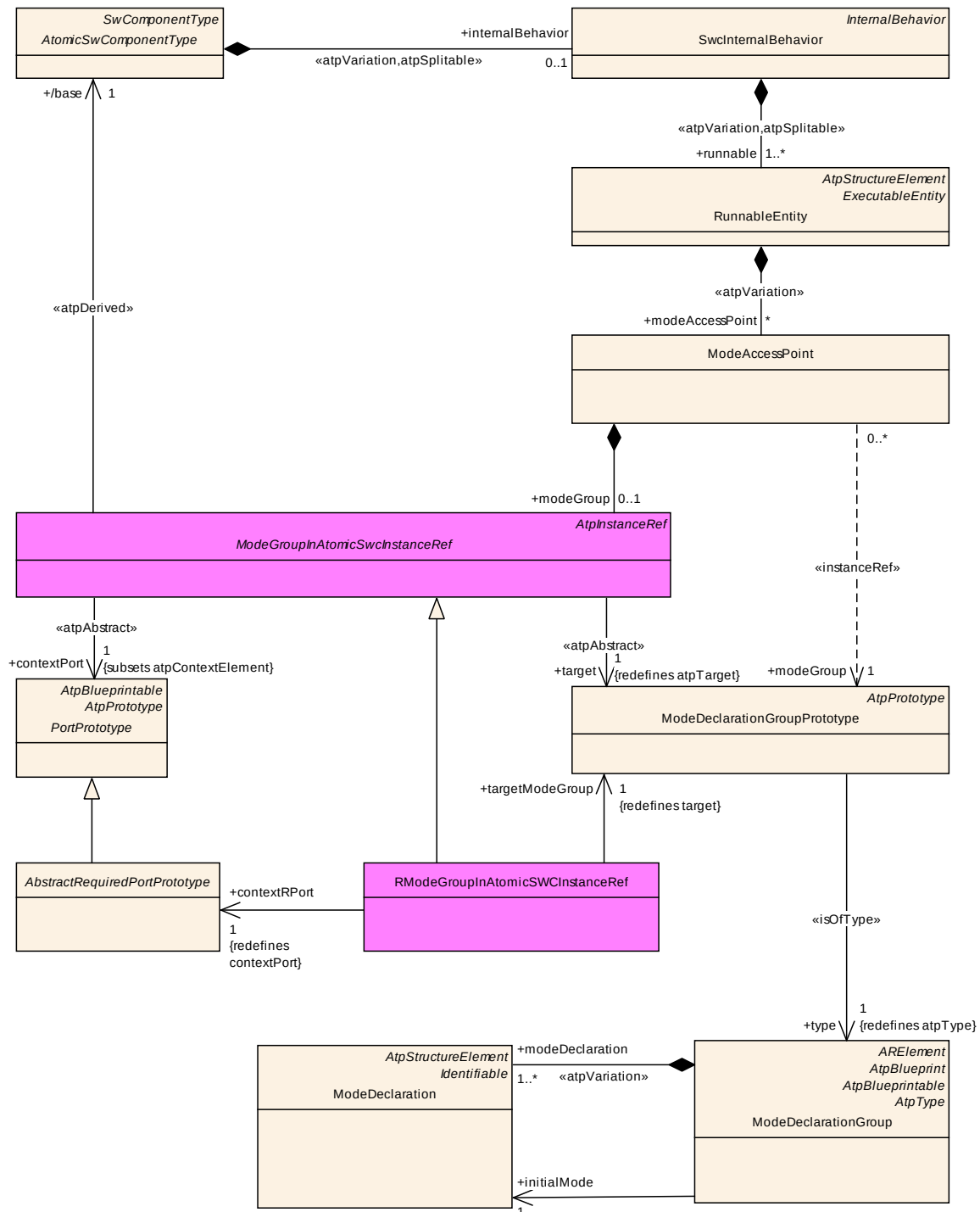


Figure D.25: Modeling of references to required `ModeDeclarationGroupPrototype` in the context of an `AtomicSwComponentType`

Class	RModeGroupInAtomicSWCInstanceRef			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components::InstanceRefs			
Note				
Base	ARObject, AtpInstanceRef, ModeGroupInAtomicSwcInstanceRef			
Attribute	Datatype	Mul.	Kind	Note

Attribute	Datatype	Mul.	Kind	Note
contextRP ort	AbstractRequire dPortPrototype	1	ref	Tags: xml.sequenceOffset=20
targetMod eGroup	ModeDeclaratio nGroupPrototyp e	1	ref	Tags: xml.sequenceOffset=30

Table D.25: RModeGroupInAtomicSWCInstanceRef

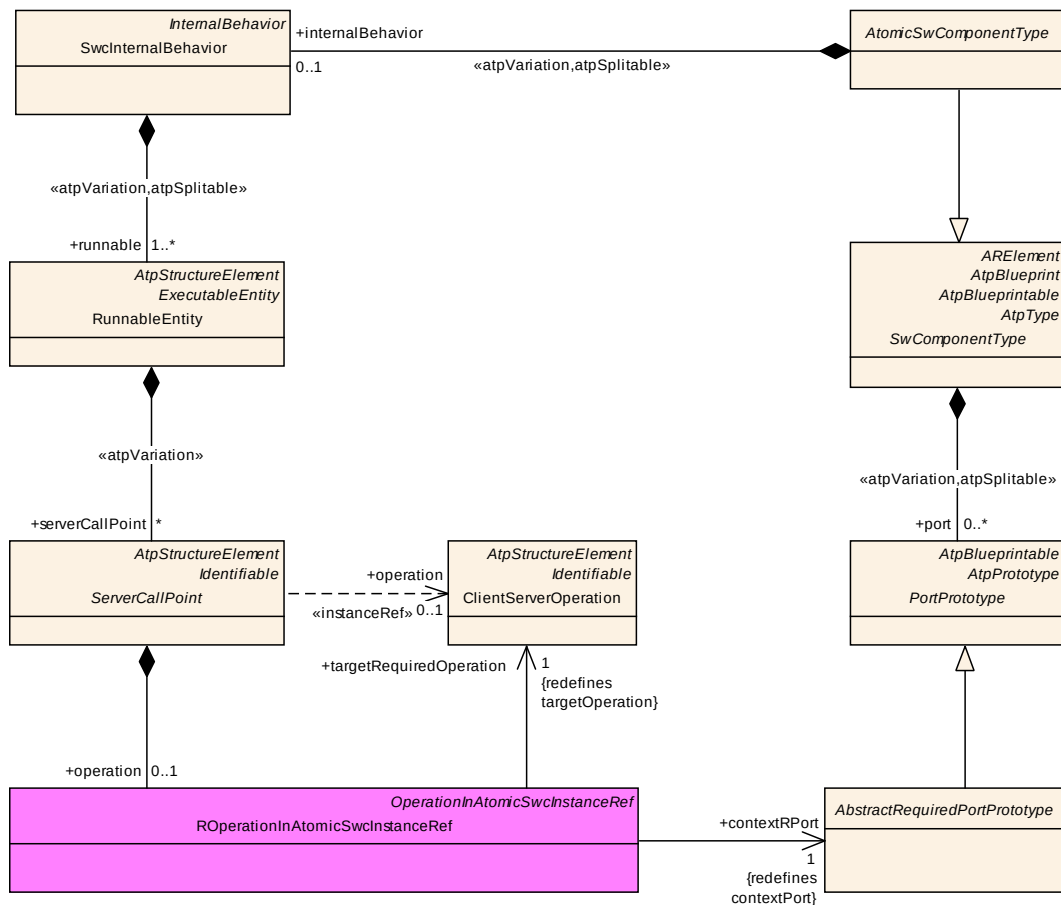


Figure D.26: Modeling of references to required `ClientServerOperation` in the context of a `SwComponentType`

Class	ROperationInAtomicSwcInstanceRef			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components::InstanceRefs			
Note				
Base	ARObject,AtplInstanceRef,OperationInAtomicSwcInstanceRef			
Attribute	Datatype	Mul.	Kind	Note
contextRP ort	AbstractRequire dPortPrototype	1	ref	Tags: xml.sequenceOffset=20
targetRequ iredOperati on	ClientServerOp eration	1	ref	Tags: xml.sequenceOffset=30

Table D.26: ROperationInAtomicSwcInstanceRef

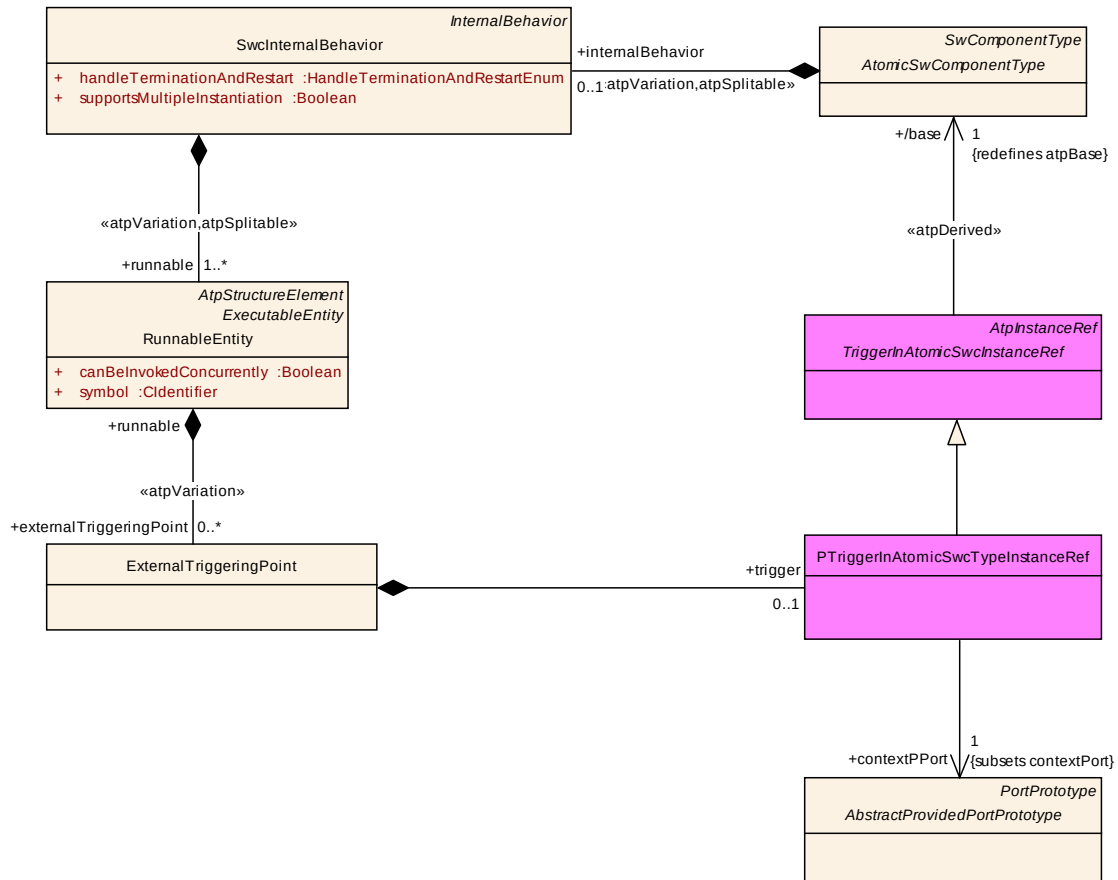


Figure D.27: Modeling of references to a **Trigger** in the context of a **SwComponentType**

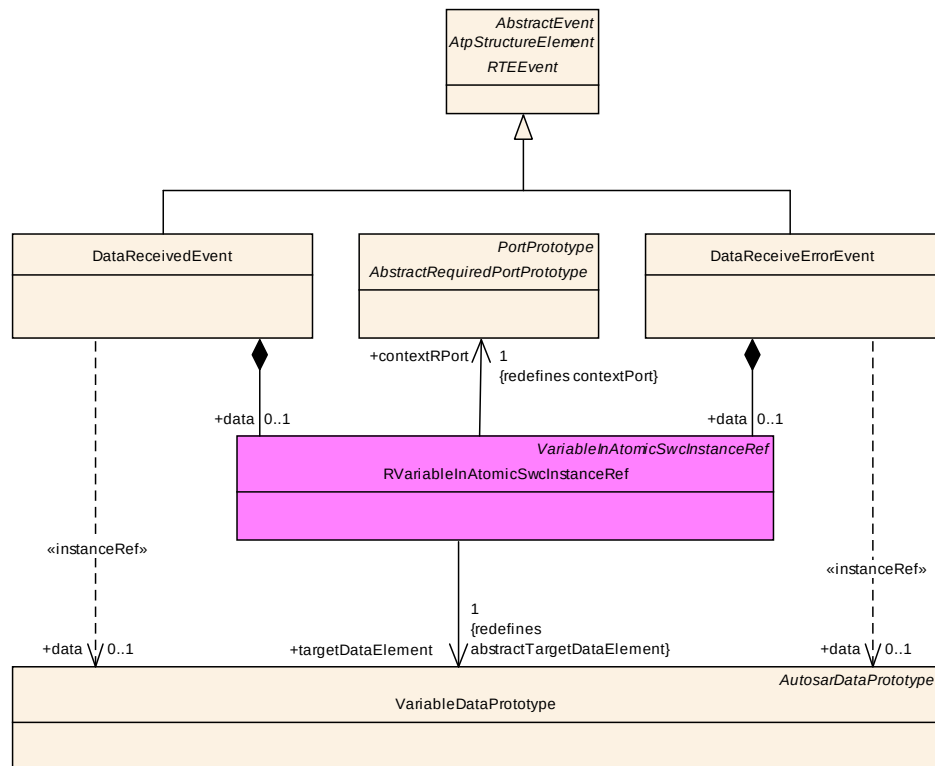


Figure D.28: Modeling of references to a **VariableDataPrototype used in the context of an **RTEEvent****

Class	RVariableInAtomicSwcInstanceRef			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components::InstanceRefs			
Note				
Base	ARObject, AtpInstanceRef, VariableInAtomicSwcInstanceRef			
Attribute	Datatype	Mul.	Kind	Note
contextRPort	AbstractRequiredPortPrototype	1	ref	Tags: xml.sequenceOffset=20
targetDataElement	VariableDataPrototype	1	ref	Tags: xml.sequenceOffset=30

Table D.27: RVariableInAtomicSwcInstanceRef

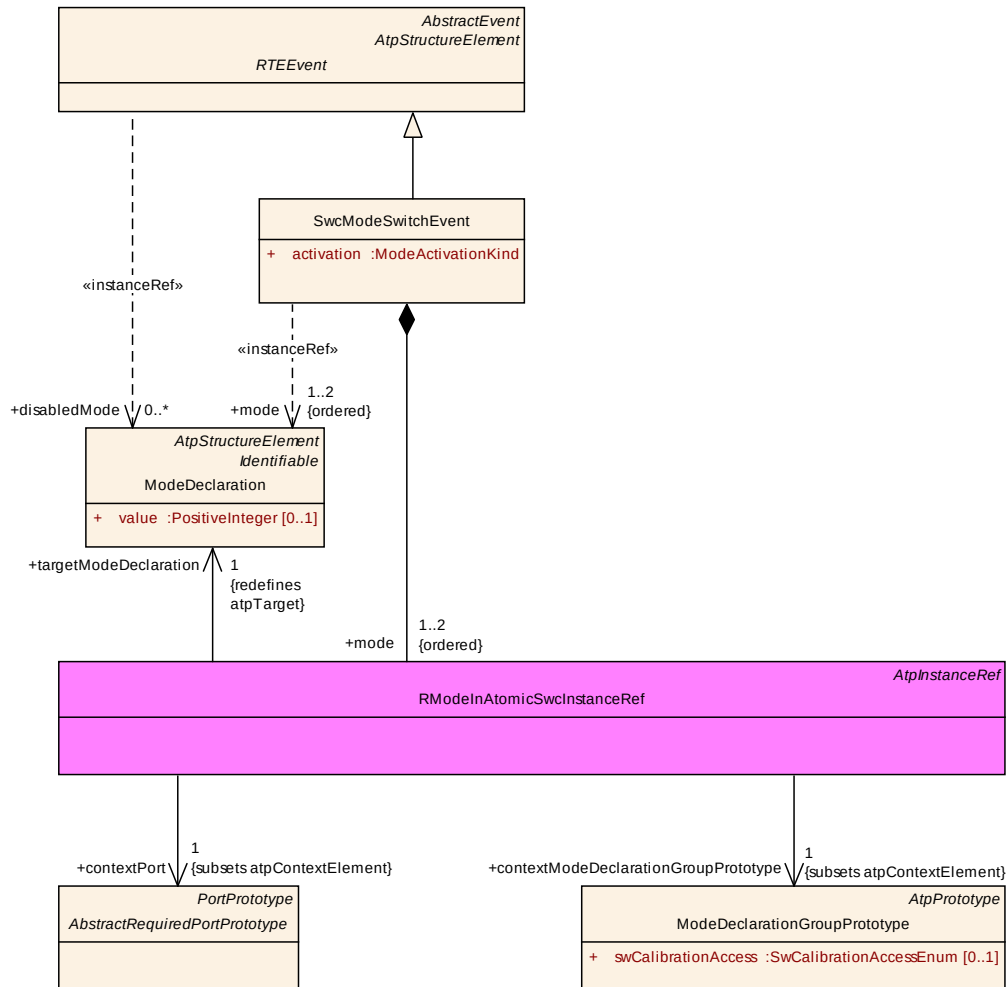


Figure D.29: Modeling of references to a **ModeDeclaration used in the context of an **RTEEvent****

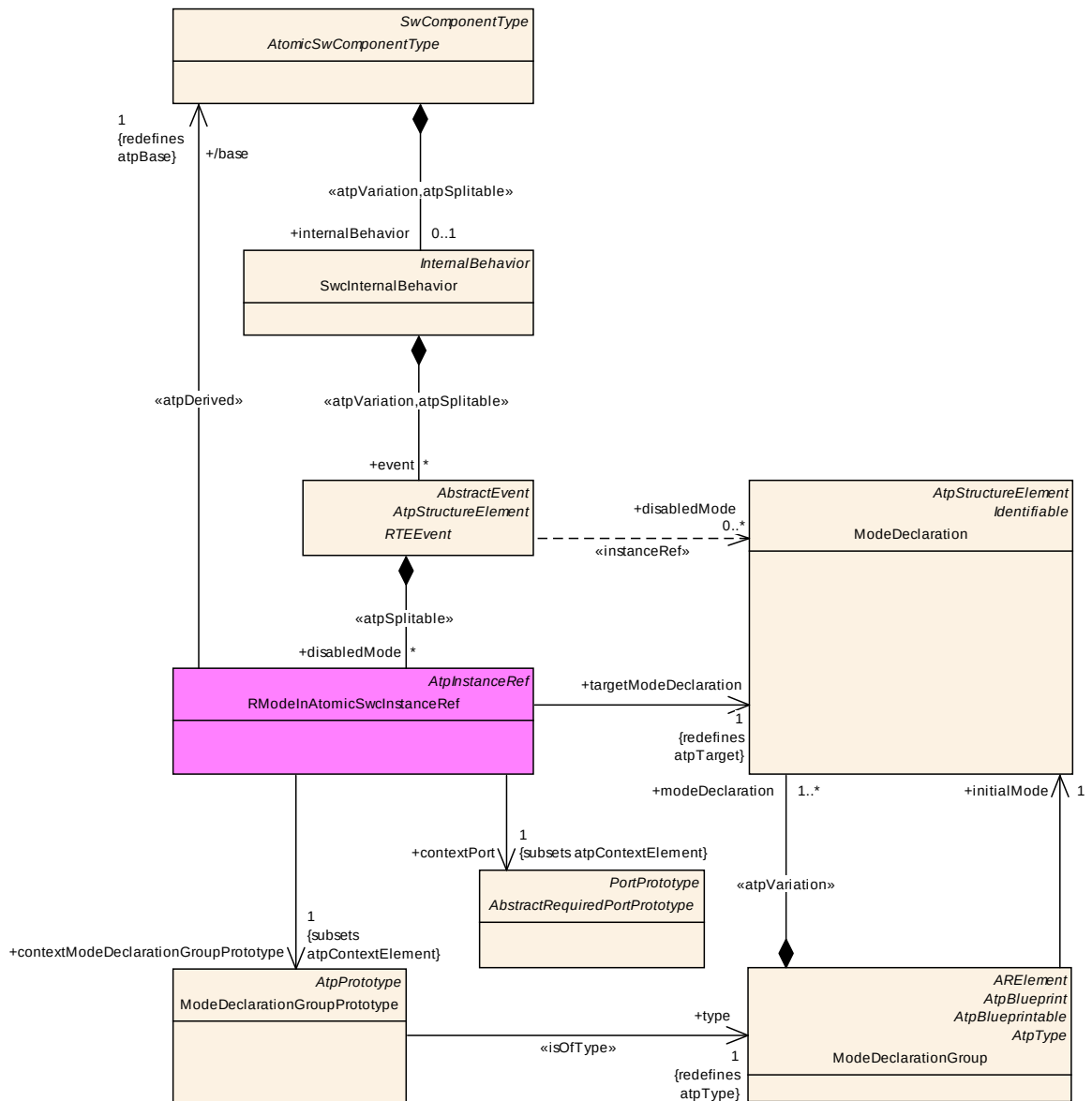


Figure D.30: Modeling of mode disabling

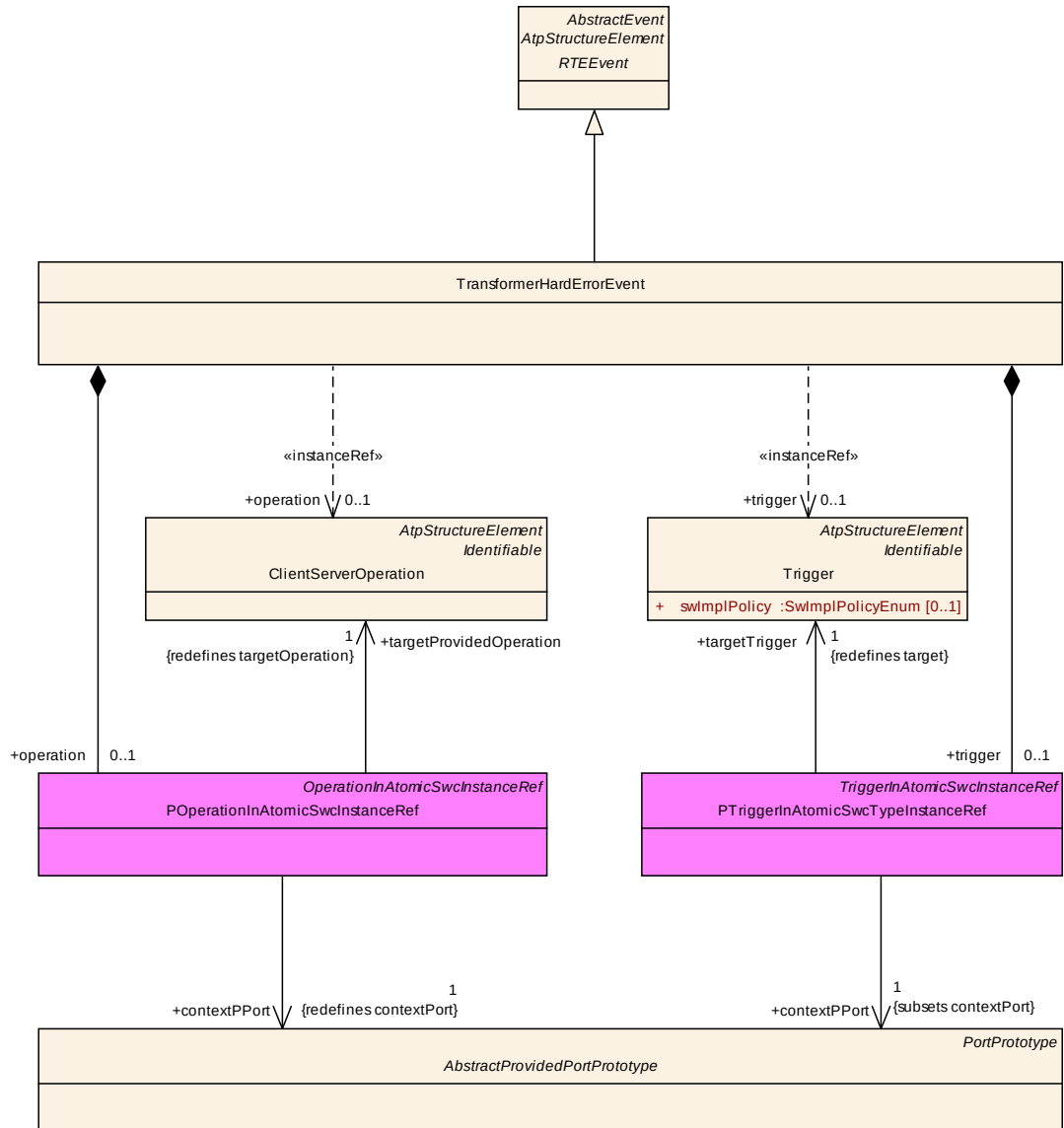


Figure D.31: Modeling of transformer error response

E Examples

E.1 Examples for the Definition of variable-size Arrays

This chapter contains some examples for the usage of variable-size arrays and its four defined use cases: Linear, square, rectangular and fully flexible.

All of these examples can be realized only using [ImplementationDataTypes](#) or, as shown here, starting with the definition of [ApplicationDataTypes](#) with corresponding [ImplementationDataTypes](#).

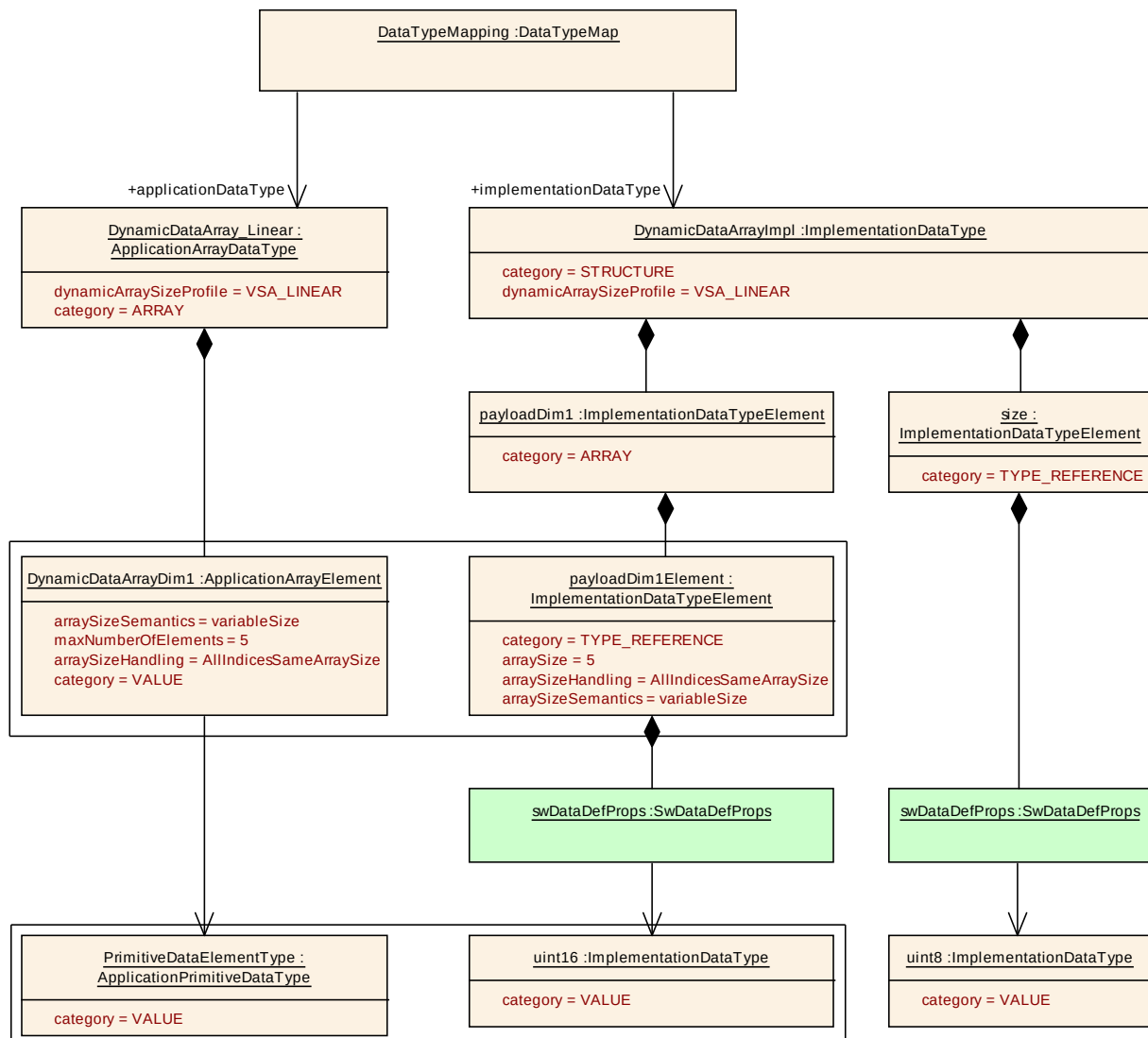


Figure E.1: Example of a linear variable size array

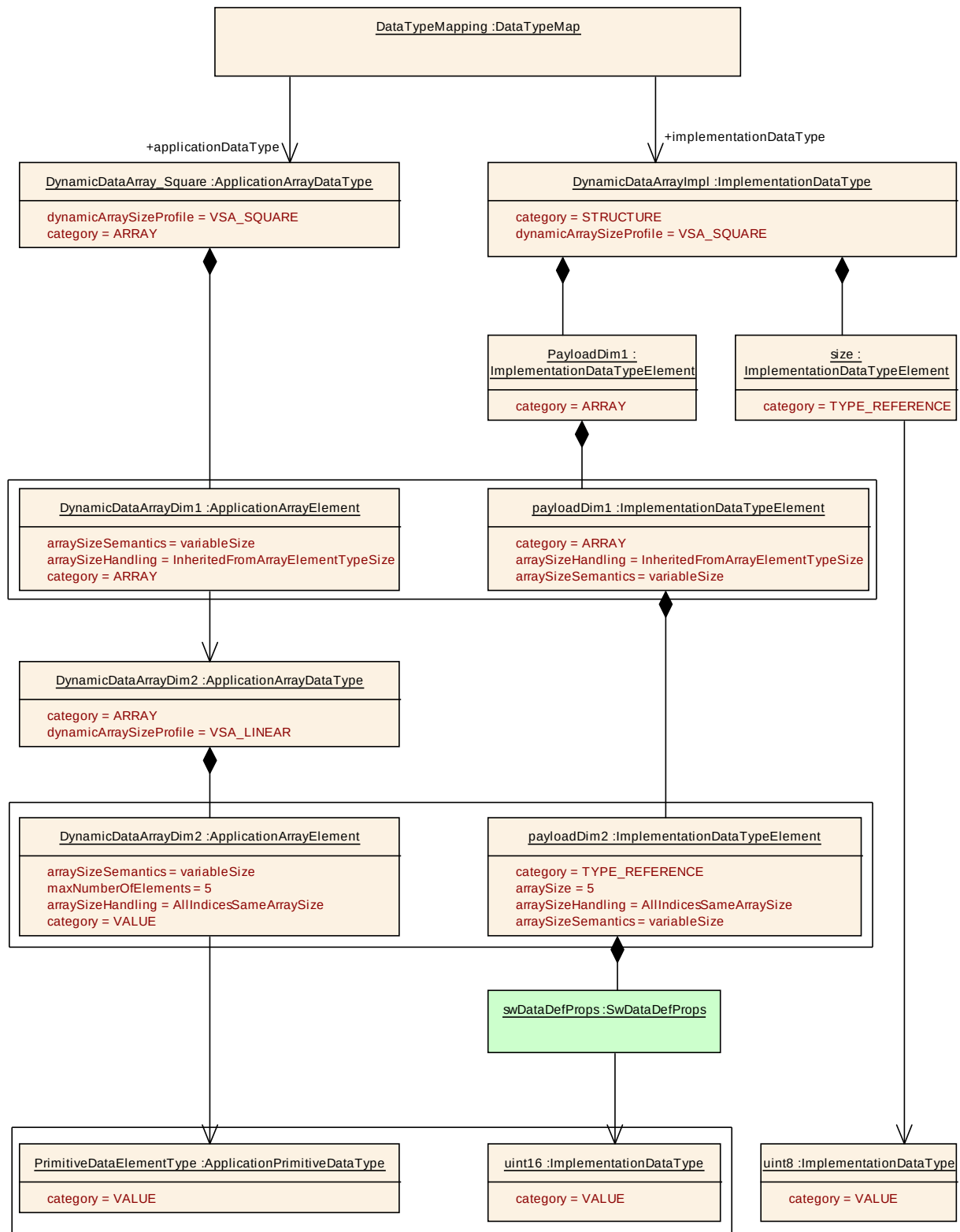


Figure E.2: Example of a square variable size array

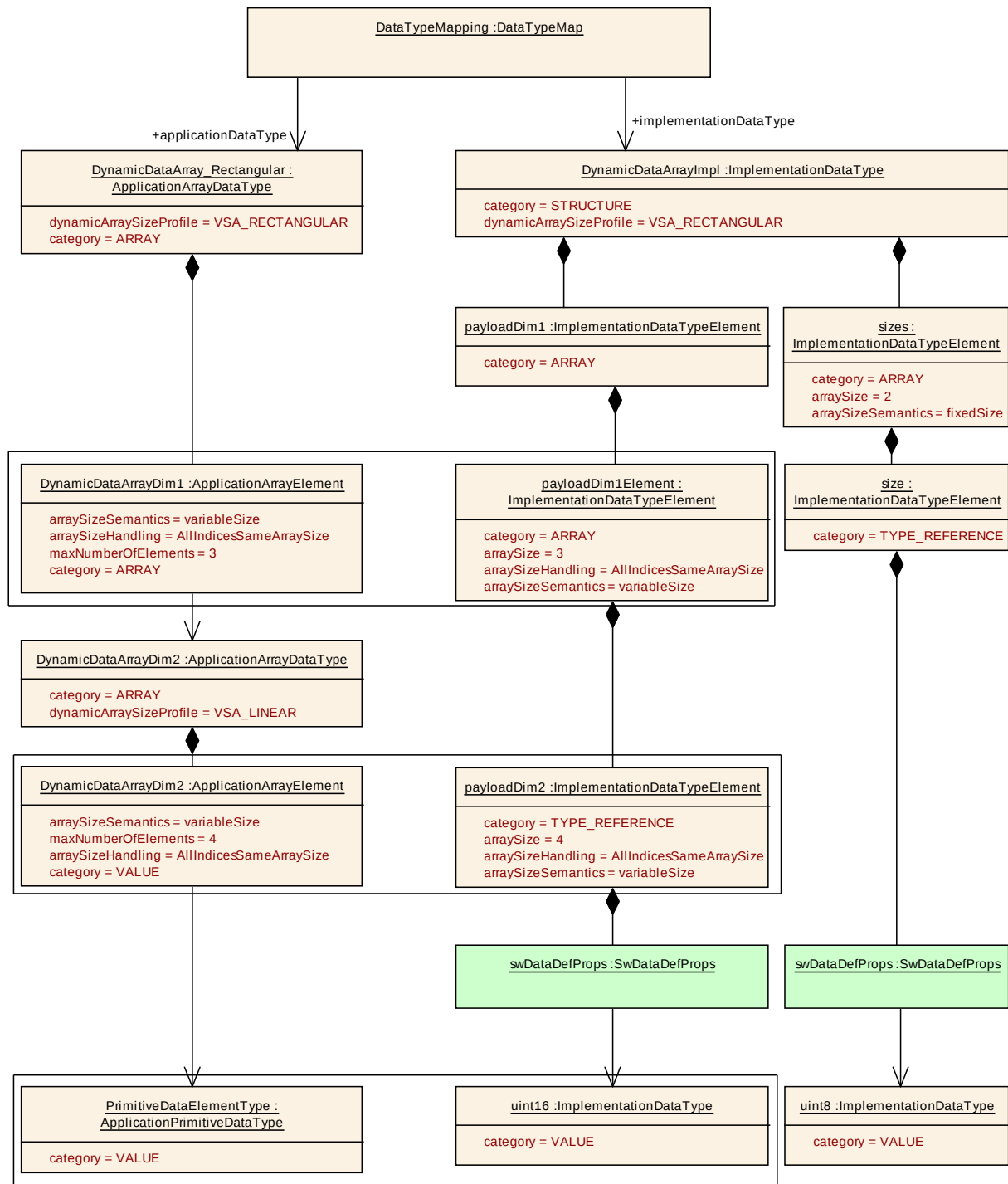


Figure E.3: Example of a rectangular variable size array

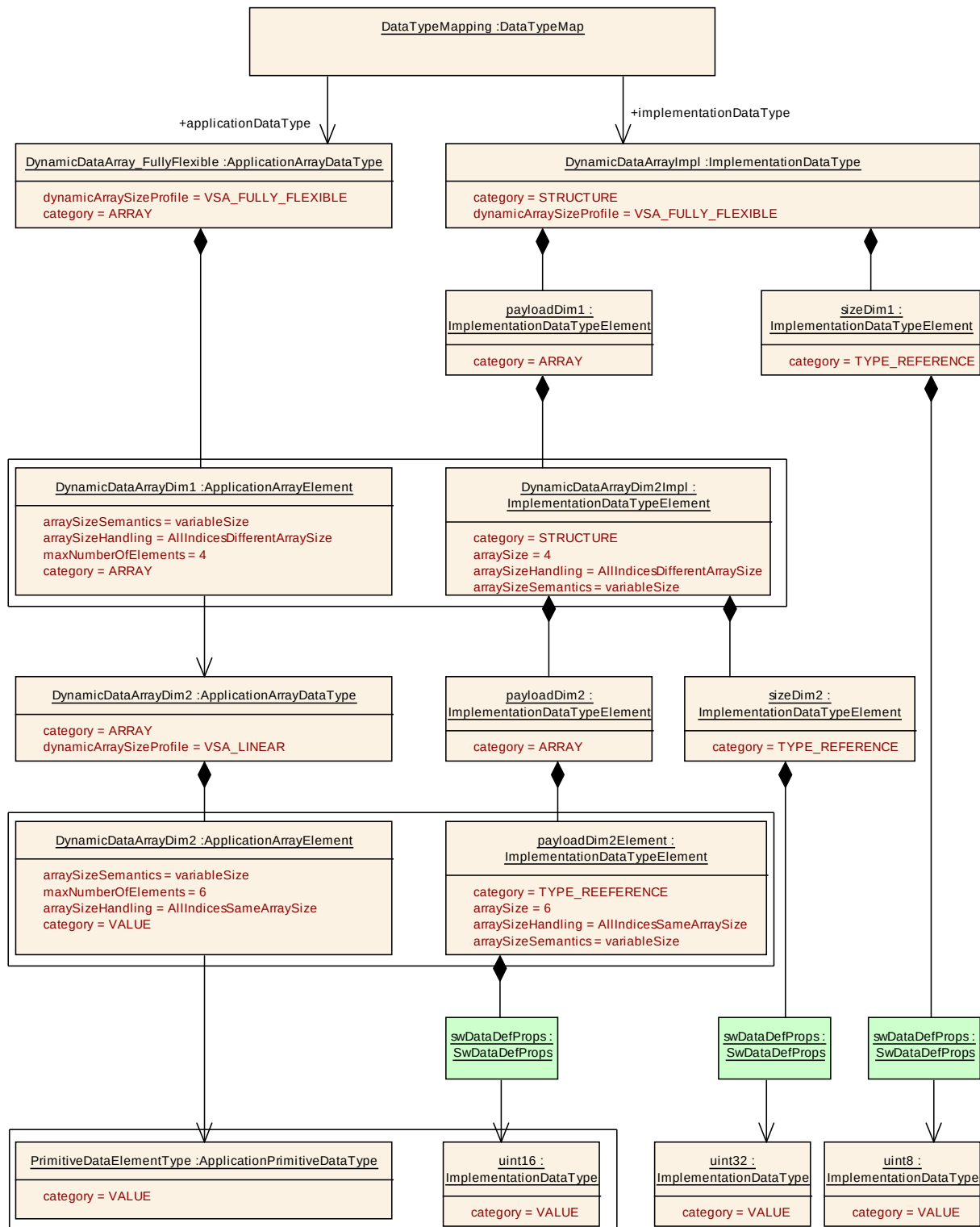


Figure E.4: Example of a fully flexible variable size array

F 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)			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::ARPackage			
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			
Attribute	Datatype	Mul.	Kind	Note
–	–	–	–	–

Table F.1: ARElement

Class	ARPackage			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::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,AtpBlueprintable,CollectableElement,Identifiable,MultilanguageReferrable,Referrable			
Attribute	Datatype	Mul.	Kind	Note
arPackage	ARPackage	*	aggr	This represents a sub package within an ARPackage, thus allowing for an unlimited package hierarchy. Stereotypes: atpSplitable; atpVariation Tags: atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=blueprintDerivationTime xml.sequenceOffset=30
element	PackageableElement	*	aggr	Elements that are part of this package Stereotypes: atpSplitable; atpVariation Tags: atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=systemDesignTime xml.sequenceOffset=20

Attribute	Datatype	Mul.	Kind	Note
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. Stereotypes: atpSplitable Tags: atp.Splitkey=shortLabel xml.sequenceOffset=10

Table F.2: ARPackage

Class	«atpMixedString» AbstractNumericalVariationPoint (abstract)			
Package	M2::AUTOSARTemplates::GenericStructure::VariantHandling::AttributeValueVariationPoints			
Note	This is an abstract NumericalValueVariationPoint. It is introduced to support the case that additional attributes are required for particular purposes.			
Base	ARObject, AttributeValueVariationPoint , FormulaExpression, SwSystemconstDependentFormula			
Attribute	Datatype	Mul.	Kind	Note
—	—	—	—	—

Table F.3: AbstractNumericalVariationPoint

Class	AdminData			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::AdminData			
Note	AdminData represents the ability to express administrative information for an element. This administration information is to be treated as meta-data such as revision id or state of the file. There are basically four kinds of meta-data • The language and/or used languages. • Revision information covering e.g. revision number, state, release date, changes. Note that this information can be given in general as well as related to a particular company. • Document meta-data specific for a company			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
docRevision (ordered)	DocRevision	*	aggr	This allows to denote information about the current revision of the object. Note that information about previous revisions can also be logged here. The entries shall be sorted descendant by date in order to reflect the history. Therefore the most recent entry representing the current version is denoted first. Tags: xml.roleElement=true; xml.roleWrapperElement=true; xml.sequenceOffset=50; xml.typeElement=false; xml.typeWrapperElement=false

Attribute	Datatype	Mul.	Kind	Note
language	LEnum	0..1	attr	This attribute specifies the master language of the document or the document fragment. The master language is the one in which the document is maintained and from which the other languages are derived from. In particular in case of inconsistencies, the information in the master language is priority. Tags: xml.sequenceOffset=20
sdg	Sdg	*	aggr	This property allows to keep special data which is not represented by the standard model. It can be utilized to keep e.g. tool specific data. Tags: xml.roleElement=true; xml.roleWrapperElement=true; xml.sequenceOffset=60; xml.typeElement=false; xml.typeWrapperElement=false
usedLanguages	MultiLanguagePlainText	0..1	aggr	This property specifies the languages which are provided in the document. Therefore it should only be specified in the top level admin data. For each language provided in the document there is one entry in MultiLanguagePlainText. The content of each entry can be used for illustration of the language. The used language itself depends on the language attribute in the entry. Tags: xml.sequenceOffset=30

Table F.4: AdminData

Class	AnyInstanceRef			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::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			
Attribute	Datatype	Mul.	Kind	Note
base	AtpClassifier	1	ref	This is the base from which navigation path begins. Stereotypes: atpDerived
contextElement	AtpFeature	*	ref	This is one step in the navigation path specified by the instance ref.
target	AtpFeature	1	ref	This is the target of the instance ref.

Table F.5: AnyInstanceRef

Class	ApplicationCompositeElementInPortInterfaceInstanceRef			
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface::InstanceRefs			
Note				
Base	ARObject, AtpInstanceRef			
Attribute	Datatype	Mul.	Kind	Note
base	DataInterface	1	ref	This represents the SenderReceiverInterface that acts as the base in this InstanceRef definition Stereotypes: atpDerived Tags: xml.sequenceOffset=10
contextDataPrototype	ApplicationCompositeElementDataPrototype	*	ref	This represents a context ApplicationCompositeDataPrototype Tags: xml.sequenceOffset=20
rootDataPrototype	AutosarDataPrototype	1	ref	This refers to the dataPrototype which is typed by theApplicationDatatype in which which the target can be found. Tags: xml.sequenceOffset=15
targetDataPrototype	ApplicationCompositeElementDataPrototype	1	ref	This represents the referenced ApplicationCompositeDataPrototype. Tags: xml.sequenceOffset=30

Table F.6: ApplicationCompositeElementInPortInterfaceInstanceRef

Class	AtpBlueprint (abstract)			
Package	M2::AUTOSARTemplates::StandardizationTemplate::AbstractBlueprintStructure			
Note	This meta-class represents the ability to act as a Blueprint. As this class is an abstract one, particular blueprint meta-classes inherit from this one.			
Base	ARObject, Identifiable , MultilanguageReferrable, Referrable			
Attribute	Datatype	Mul.	Kind	Note
blueprintPolicy	BlueprintPolicy	*	aggr	This role indicates whether the blueprintable element will be modifiable or not motifiable.

Table F.7: AtpBlueprint

Class	AtpInstanceRef (abstract)			
Package	M2::AUTOSARTemplates::GenericStructure::AbstractStructure			
Note	An M0 instance of a classifier may be represented as a tree rooted at that instance, where under each node come the sub-trees representing the instances which act as features under that node. An instance ref specifies a navigation path from any M0 tree-instance of the base (which is a classifier) to a leaf (which is an instance of the target).			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note

Attribute	Datatype	Mul.	Kind	Note
atpBase	AtpClassifier	1	ref	This is the base from which the navigation path starts. Stereotypes: atpAbstract; atpDerived
atpContextElement (ordered)	AtpPrototype	*	ref	This is one particular step in the navigation path. Stereotypes: atpAbstract
atpTarget	AtpFeature	1	ref	This is the target of the instance ref. In other words it is the terminal of the navigation path. Stereotypes: atpAbstract

Table F.8: AtpInstanceRef

Class	«atpMixedString» AttributeValueVariationPoint (abstract)			
Package	M2::AUTOSARTemplates::GenericStructure::VariantHandling::AttributeValueVariationPoints			
Note	This class represents the ability to derive the value of the Attribute from a system constant (by SwSystemconstDependentFormula). It also provides a bindingTime.			
Base	ARObject,FormulaExpression, SwSystemconstDependentFormula			
Attribute	Datatype	Mul.	Kind	Note
bindingTime	BindingTimeEnum	0..1	attr	This is the binding time in which the attribute value needs to be bound. If this attribute is missing, the attribute is not a variation point. In particular this means that it needs to be a single value according to the type specified in the pure model. It is an error if it is still a formula. Tags: xml.attribute=true
blueprintValue	String	0..1	attr	This represents a description that documents how the value shall be defined when deriving objects from the blueprint. Tags: xml.attribute=true
sd	String	0..1	attr	This special data is provided to allow synchronization of Attribute value variation points with variant management systems. The usage is subject of agreement between the involved parties. Tags: xml.attribute=true
shortLabel	PrimitiveIdentifier	0..1	attr	This allows to identify the variation point. It is also intended to allow RTE support for CompileTime Variation points. Tags: xml.attribute=true

Table F.9: AttributeValueVariationPoint

Enumeration	BindingTimeEnum
Package	M2::AUTOSARTemplates::GenericStructure::VariantHandling
Note	This enumerator specifies the applicable binding times for the pre build variation points.
Literal	Description
codeGenerationTime	• Coding by hand, based on requirements document. • Tool based code generation, e.g. from a model. • The model may contain variants. • Only code for the selected variant(s) is actually generated.
linkTime	Configure what is included in object code, and what is omitted Based on which variant(s) are selected E.g. for modules that are delivered as object code (as opposed to those that are delivered as source code)
preCompileTime	This is typically the C-Preprocessor. Exclude parts of the code from the compilation process, e.g., because they are not required for the selected variant, because they are incompatible with the selected variant, because they require resources that are not present in the selected variant. Object code is only generated for the selected variant(s). The code that is excluded at this stage code will not be available at later stages.
systemDesignTime	• Designing the VFB. • Software Component types (PortInterfaces). • SWC Prototypes and the Connections between SWCprototypes. • Designing the Topology • ECUs and interconnecting Networks • Designing the Communication Matrix and Data Mapping

Table F.10: BindingTimeEnum

Class	BswImplementation			
Package	M2::AUTOSARTemplates::BswModuleTemplate::BswImplementation			
Note	Contains the implementation specific information in addition to the generic specification (BswModuleDescription and BswBehavior). It is possible to have several different BswImplementations referring to the same BswBehavior. Tags: atp.recommendedPackage=BswImplementations			
Base	ARElement , ARObject , CollectableElement , Identifiable , Implementation , MultilanguageReferrable , PackageableElement , Referrable			
Attribute	Datatype	Mul.	Kind	Note
arReleaseVersion	RevisionLabelString	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.

Attribute	Datatype	Mul.	Kind	Note
behavior	BswInternalBehavior	1	ref	The behavior of this implementation.
debugInfo	BswDebugInfo	0..1	aggr	Collects the debug info for this implementation. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime
preconfiguredConfiguration	EcucModuleConfigurationValues	*	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: xml.roleWrapperElement=true
recommendedConfiguration	EcucModuleConfigurationValues	*	ref	Reference to one or more sets of recommended configuration values for this module or module cluster.
vendorApiInfix	Identifier	0..1	ref	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: <ModuleName>_<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.

Attribute	Datatype	Mul.	Kind	Note
vendorSpecificModuleDef	EcucModuleDef	*	ref	Reference to - the vendor specific EcucModuleDef used in this BswImplementation if it represents a single module - several EcucModuleDefs used in this BswImplementation if it represents a cluster of modules - one or no EcucModuleDefs used in this BswImplementation if it represents a library Tags: xml.roleWrapperElement=true

Table F.11: BswImplementation

Class	BswModuleDescription			
Package	M2::AUTOSARTemplates::BswModuleTemplate::BswOverview			
Note	Root element for the description of a single BSW module or BSW cluster. In case it describes a BSW module, the short name of this element equals the name of the BSW module. Tags: atp.recommendedPackage=BswModuleDescriptions			
Base	ARElement , ARObject , AtpBlueprint , AtpBlueprintable , AtpClassifier , AtpFeature , AtpStructureElement , CollectableElement , Identifiable , Multilanguage , Referrable , PackageableElement , Referrable			
Attribute	Datatype	Mul.	Kind	Note
bswModuleDependency	BswModuleDependency	*	aggr	Describes the dependency to another BSW module. Stereotypes: atpSplitable; atpVariation Tags: atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=20
bswModuleDocumentation	SwComponentDocumentation	0..1	aggr	This adds a documentation to the BSW module. Stereotypes: atpSplitable; atpVariation Tags: atp.Splitkey=bswModuleDocumentation, variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=6
internalBehavior	BswInternalBehavior	*	aggr	The various BswInternalBehaviors associated with a BswModuleDescription can be distributed over several physical files. Therefore the aggregation is «atpSplitable». Stereotypes: atpSplitable Tags: atp.Splitkey=shortName xml.sequenceOffset=65

Attribute	Datatype	Mul.	Kind	Note
moduleId	PositiveInteger	0..1	attr	Refers to the BSW Module Identifier defined by the AUTOSAR standard. For non-standardized modules, a proprietary identifier can be optionally chosen. Tags: xml.sequenceOffset=5
outgoingCallback	BswModuleEntry	*	ref	Specifies a callback, which will be called from this module if required by another module. Stereotypes: atpSplitable; atpVariation Tags: atp.Splitkey=outgoingCallback, variation Point.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=15
providedClientServerEntry	BswModuleClientServerEntry	*	aggr	Specifies that this module provides a client server entry which can be called from another partition or core. This entry is declared locally to this context and will be connected to the requiredClientServerEntry of another or the same module via the configuration of the BSW Scheduler. Stereotypes: atpSplitable; atpVariation Tags: atp.Splitkey=shortName, variation Point.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=45
providedData	VariableDataPrototype	*	aggr	Specifies a data prototype provided by this module in order to be read from another partition or core. The providedData is declared locally to this context and will be connected to the requiredData of another or the same module via the configuration of the BSW Scheduler. Stereotypes: atpSplitable; atpVariation Tags: atp.Splitkey=shortName, variation Point.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=55
providedEntry	BswModuleEntry	*	ref	Specifies an entry provided by this module which can be called by other modules. This includes "main" functions and interrupt routines, but not callbacks (because the signature of a callback is defined by the caller). Stereotypes: atpSplitable; atpVariation Tags: atp.Splitkey=providedEntry, variation Point.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=10

Attribute	Datatype	Mul.	Kind	Note
providedModeGroup	ModeDeclarationGroupPrototype	*	aggr	A set of modes which is owned and provided by this module or cluster. It can be connected to the requiredModeGroups of other modules or clusters via the configuration of the BswScheduler. It can also be synchronized with modes provided via ports by an associated ServiceSwComponentType, EcuAbstractionSwComponentType or ComplexDeviceDriverSwComponentType. Stereotypes: atpSplitable; atpVariation Tags: atp.Splitkey=shortName, variation Point.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=25
releasedTrigger	Trigger	*	aggr	A Trigger released by this module or cluster. It can be connected to the requiredTriggers of other modules or clusters via the configuration of the BswScheduler. It can also be synchronized with Triggers provided via ports by an associated ServiceSwComponentType, EcuAbstractionSwComponentType or ComplexDeviceDriverSwComponentType. Stereotypes: atpSplitable; atpVariation Tags: atp.Splitkey=shortName, variation Point.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=35
requiredClientServerEntry	BswModuleClientServerEntry	*	aggr	Specifies that this module requires a client server entry which can be implemented on another partition or core. This entry is declared locally to this context and will be connected to the providedClientServerEntry of another or the same module via the configuration of the BSW Scheduler. Stereotypes: atpSplitable; atpVariation Tags: atp.Splitkey=shortName, variation Point.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=50
requiredData	VariableDataPrototype	*	aggr	Specifies a data prototype required by this module in order to be provided from another partition or core. The requiredData is declared locally to this context and will be connected to the providedData of another or the same module via the configuration of the BswScheduler. Stereotypes: atpSplitable; atpVariation Tags: atp.Splitkey=shortName, variation Point.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=60

Attribute	Datatype	Mul.	Kind	Note
requiredModeGroup	ModeDeclarationGroupPrototype	*	aggr	Specifies that this module or cluster depends on a certain mode group. The requiredModeGroup is local to this context and will be connected to the providedModeGroup of another module or cluster via the configuration of the BswScheduler. Stereotypes: atpSplitable; atpVariation Tags: atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=30
requiredTrigger	Trigger	*	aggr	Specifies that this module or cluster reacts upon an external trigger. This requiredTrigger is declared locally to this context and will be connected to the providedTrigger of another module or cluster via the configuration of the BswScheduler. Stereotypes: atpSplitable; atpVariation Tags: atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=40

Table F.12: BswModuleDescription

Class	BswModuleEntry			
Package	M2::AUTOSARTemplates::BswModuleTemplate::BswInterfaces			
Note	This class represents a single API entry (C-function prototype) into the BSW module or cluster. The name of the C-function is equal to the short name of this element with one exception: In case of multiple instances of a module on the same CPU, special rules for "infixes" apply, see description of class BswImplementation. Tags: atp.recommendedPackage=BswModuleEntrys			
Base	ARElement , ARObject , AtpBlueprint , AtpBlueprintable , CollectableElement , Identifiable , MultilanguageReferrable , PackageableElement , Referrable			
Attribute	Datatype	Mul.	Kind	Note
argument (ordered)	SwServiceArg	*	aggr	An argument belonging to this BswModuleEntry. Stereotypes: atpVariation Tags: vh.latestBindingTime=blueprintDerivationTime xml.sequenceOffset=45
callType	BswCallType	1	attr	The type of call associated with this service. Tags: xml.sequenceOffset=25

Attribute	Datatype	Mul.	Kind	Note
executionContext	BswExecutionContext	1	attr	Specifies the execution context which is required (in case of entries into this module) or guaranteed (in case of entries called from this module) for this service. Tags: xml.sequenceOffset=30
isReentrant	Boolean	1	attr	Reentrancy from the viewpoint of function callers: • True: Enables the service to be invoked again, before the service has finished. • False: It is prohibited to invoke the service again before it has finished. Tags: xml.sequenceOffset=15
isSynchronous	Boolean	1	attr	Synchronicity from the viewpoint of function callers: • True: This calls a synchronous service, i.e. the service is completed when the call returns. • False: The service (on semantical level) may not be complete when the call returns. Tags: xml.sequenceOffset=20
returnType	SwServiceArg	0..1	aggr	The return type belonging to this bswModuleEntry. Tags: xml.sequenceOffset=40
role	Identifier	0..1	ref	Specifies the role of the entry in the given context. It shall be equal to the standardized name of the service call, especially in cases where no ServiceIdentifier is specified, e.g. for callbacks. Note that the ShortName is not always sufficient because it may be vendor specific (e.g. for callbacks which can have more than one instance). Tags: xml.sequenceOffset=10
serviceId	PositiveInteger	0..1	attr	Refers to the service identifier of the Standardized Interfaces of AUTOSAR basic software. For non-standardized interfaces, it can optionally be used for proprietary identification. Tags: xml.sequenceOffset=5
swServiceImplementationPolicy	SwServiceImplementationPolicyEnum	1	attr	Denotes the implementation policy as a standard function call, inline function or macro. This has to be specified on interface level because it determines the signature of the call. Tags: xml.sequenceOffset=35

Table F.13: BswModuleEntry

Class	BswSchedulableEntity			
Package	M2::AUTOSARTemplates::BswModuleTemplate::BswBehavior			
Note	BSW module entity, which is designed for control by the BSW Scheduler. It may for example implement a so-called "main" function.			
Base	ARObject, BswModuleEntity, ExecutableEntity, Identifiable, Multilanguage Referrable, Referrable			
Attribute	Datatype	Mul.	Kind	Note
–	–	–	–	–

Table F.14: BswSchedulableEntity

Class	BswServiceDependency			
Package	M2::AUTOSARTemplates::BswModuleTemplate::BswBehavior			
Note	Specialization of ServiceDependency in the context of an BswInternalBehavior. It allows to associate BswModuleEntries and data defined for a BSW module or cluster to a given ServiceNeeds element.			
Base	ARObject, ServiceDependency			
Attribute	Datatype	Mul.	Kind	Note
assignedData	RoleBasedData Assignment	*	aggr	Defines the role of an associated data object (owned by this module or cluster) in the context of the ServiceNeeds element. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime
assignedEntryRole	RoleBasedBswModuleEntryAssignment	*	aggr	Defines the role of an associated BswModuleEntry in the context of the ServiceNeeds element. Stereotypes: atpSplittable; atpVariation Tags: atp.Splitkey=assignedEntryRole, variationPoint.shortLabel vh.latestBindingTime=preCompileTime
ident	BswServiceDependencyIdent	0..1	aggr	This adds the ability to become referrable to BswServiceDependency. Tags: atp.Status=shallBecomeMandatory xml.sequenceOffset=-100
serviceNeeds	ServiceNeeds	1	aggr	The associated ServiceNeeds.

Table F.15: BswServiceDependency

Enumeration	ByteOrderEnum			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::PrimitiveTypes			
Note	When more than one byte is stored in the memory the order of those bytes may differ depending on the architecture of the processing unit. If the least significant byte is stored at the lowest address, this architecture is called little endian and otherwise it is called big endian. ByteOrder is very important in case of communication between different PUs or ECUs.			

<i>Literal</i>	<i>Description</i>
mostSignificantByteFirst	Most significant byte shall come at the lowest address (also known as BigEndian or as Motorola-Format)
mostSignificantByteLast	Most significant byte shall come highest address (also known as LittleEndian or as Intel-Format)
opaque	For opaque data endianness conversion has to be configured to Opaque. See AUTOSAR COM Specification for more details.

Table F.16: ByteOrderEnum

<i>Class</i>	<i>Chapter</i>			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::Documentation::Chapters			
Note	This meta-class represents a chapter of a document. Chapters are the primary structuring element in documentation.			
Base	ARObject, DocumentViewSelectable, Identifiable , Multilanguage Referrable, Paginateable, Referrable			
<i>Attribute</i>	<i>Datatype</i>	<i>Mul.</i>	<i>Kind</i>	<i>Note</i>
chapterModel	ChapterModel	1	aggr	This represents the overall contents of the chapter. Tags: xml.roleElement=false; xml.roleWrapperElement=false; xml.typeElement=false; xml.typeWrapperElement=false
helpEntry	String	0..1	attr	This specifies an entry point in an online help system to be linked with the parent class. The syntax shall be defined by the applied help system respectively help system generator. Maybe it is a concatenated Identifier, but as of now we leave it as an arbitrary string. Tags: xml.attribute=true

Table F.17: Chapter

<i>Class</i>	<i>CompuConstFormulaContent</i>			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::ComputationMethod			
Note	This meta-class represents the fact that the constant value of the computation method is represented by a variation point. This difference is due to compatibility with ASAM HDO.			
Base	ARObject, CompuConstContent			
<i>Attribute</i>	<i>Datatype</i>	<i>Mul.</i>	<i>Kind</i>	<i>Note</i>

Attribute	Datatype	Mul.	Kind	Note
vf	Numerical	1	attr	Value calculated via a system constant. This element is included in every case where parameters should be generated from numerical values during compile time (not runtime!). Thus for example, the influence of the cylinder number on conversion formulae can be introduced in a repeatable manner. Stereotypes: atpVariation Tags: vh.latestBindingTime=codeGenerationTime xml.sequenceOffset=30

Table F.18: CompuConstFormulaContent

Class	«atpMixedString» ConditionByFormula			
Package	M2::AUTOSARTemplates::GenericStructure::VariantHandling			
Note	This class represents a condition which is computed based on system constants according to the specified expression. The expected result is considered as boolean value. The result of the expression is interpreted as a condition. • "0" represents "false"; • a value other than zero is considered "true"			
Base	ARObject,FormulaExpression, SwSystemconstDependentFormula			
Attribute	Datatype	Mul.	Kind	Note
bindingTime	BindingTimeEnum	1	attr	This attribute specifies the point in time when condition may be evaluated at earliest. At this point in time all referenced system constants shall have a value. Tags: xml.attribute=true

Table F.19: ConditionByFormula

Class	DataMapping (abstract)			
Package	M2::AUTOSARTemplates::SystemTemplate::DataMapping			
Note	Mapping of port elements (data elements and parameters) to frames and signals.			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
communicationDirection	CommunicationDirectionType	0..1	attr	This attribute controls the direction into which the mapped SystemSignal is communicated with respect to the kind of PortPrototype used as the context element of the DataMapping.
eventGroup	ConsumedEventGroup	*	ref	Via this reference a connection between the VFB View and the Ethernet EventGroups can be created.

Attribute	Datatype	Mul.	Kind	Note
eventHandler	EventHandler	*	ref	Via this reference a connection between the VFB View and the Ethernet EventHandlers can be created.
introduction	Documentation Block	0..1	aggr	This represents introductory documentation about the data mapping.
serviceInstance	AbstractServiceInstance	*	ref	Via this reference a connection between the VFB View and the Ethernet Services can be created.

Table F.20: DataMapping

Class	DataTransformation			
Package	M2::AUTOSARTemplates::SystemTemplate::Transformer			
Note	A DataTransformation represents a transformer chain. It is an ordered list of transformers.			
Base	ARObject, Identifiable , MultilanguageReferrable, Referrable			
Attribute	Datatype	Mul.	Kind	Note
executeDespiteDataUnavailability	Boolean	1	attr	Specifies whether the transformer is executed even if no input data are available.
transformerChain (ordered)	Transformation Technology	1..*	ref	

Table F.21: DataTransformation

Class	Describable (abstract)			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::Identifiable			
Note	This meta-class represents the ability to add a descriptive documentation to non identifiable elements.			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
desc	MultiLanguageOverviewParagraph	0..1	aggr	This represents a general but brief (one paragraph) description what the object in question is about. It is only one paragraph! Desc is intended to be collected into overview tables. This property helps a human reader to identify the object in question. More elaborate documentation, (in particular how the object is built or used) should go to "introduction". Tags: xml.sequenceOffset=-60
category	CategoryString	0..1	attr	The category is a keyword that specializes the semantics of the Describable. It affects the expected existence of attributes and the applicability of constraints. Tags: xml.sequenceOffset=-50

Attribute	Datatype	Mul.	Kind	Note
adminData	AdminData	0..1	aggr	This represents the administrative data for the describable object. Tags: xml.sequenceOffset=-20
introduction	DocumentationBlock	0..1	aggr	This represents more information about how the object in question is built or is used. Therefore it is a DocumentationBlock. Tags: xml.sequenceOffset=-30

Table F.22: Describable

Class	«atpMixed» DocumentationBlock			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::Documentation::BlockElements			
Note	This class represents a documentation block. It is made of basic text structure elements which can be displayed in a table cell.			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
defList	DefList	0..1	aggr	This represents a definition list in the documentation block. Stereotypes: atpVariation Tags: vh.latestBindingTime=postBuild xml.sequenceOffset=40
figure	MIFigure	0..1	aggr	This represents a figure in the documentation block. Stereotypes: atpVariation Tags: vh.latestBindingTime=postBuild xml.sequenceOffset=70
formula	MIFormula	0..1	aggr	This is a formula in the definition block. Stereotypes: atpVariation Tags: vh.latestBindingTime=postBuild xml.sequenceOffset=60
labeledList	LabeledList	0..1	aggr	This represents a labeled list. Stereotypes: atpVariation Tags: vh.latestBindingTime=postBuild xml.sequenceOffset=50
list	List	0..1	aggr	This represents numbered or unnumbered list. Stereotypes: atpVariation Tags: vh.latestBindingTime=postBuild xml.sequenceOffset=30
msrQueryP2	MsrQueryP2	0..1	aggr	

Attribute	Datatype	Mul.	Kind	Note
note	Note	0..1	aggr	This represents a note in the text flow. Stereotypes: atpVariation Tags: vh.latestBindingTime=postBuild xml.sequenceOffset=80
p	MultiLanguageParagraph	0..1	aggr	This is one particular paragraph. Stereotypes: atpVariation Tags: vh.latestBindingTime=postBuild xml.sequenceOffset=10
structured Req	StructuredReq	0..1	aggr	This aggregation supports structured requirements embedded in a documentation block. Stereotypes: atpVariation Tags: vh.latestBindingTime=postBuild xml.sequenceOffset=100
trace	TraceableText	0..1	aggr	This represents traceable text in the documentation block. This allows to specify requirements/constraints in any documentation block. The kind of the trace is specified in the category. Stereotypes: atpVariation Tags: vh.latestBindingTime=postBuild xml.sequenceOffset=90
verbatim	MultiLanguageVerbatim	0..1	aggr	This represents one particular verbatim text. Stereotypes: atpVariation Tags: vh.latestBindingTime=postBuild xml.sequenceOffset=20

Table F.23: DocumentationBlock

Class	EcuInstance			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreTopology			
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,PackageableElement,Referrable			
Attribute	Datatype	Mul.	Kind	Note

Attribute	Datatype	Mul.	Kind	Note
associatedComIPduGroup	ISignalPduGroup	*	ref	With this reference it is possible to identify which ISignalPduGroups are applicable for which CommunicationConnector/ ECU. Only top level ISignalPduGroups shall be referenced by an EcuInstance. If an ISignalPduGroup contains other ISignalPduGroups than these contained ISignalPduGroups shall not be referenced by the EcuInstance. Contained ISignalPduGroups are associated to an EcuInstance via the top level ISignalPduGroup.
associatedPduRIPduGroup	PduRIPduGroup	*	ref	With this reference it is possible to identify which PduR IPdu Groups are applicable for which CommunicationConnector/ ECU.
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.
comConfigurationGwTimeBase	TimeValue	0..1	attr	The period between successive calls to Com_MainFunctionRouteSignals of the AUTOSAR COM module in seconds.
comConfigurationRxTimeBase	TimeValue	0..1	attr	The period between successive calls to Com_MainFunctionRx of the AUTOSAR COM module in seconds.
comConfigurationTxTimeBase	TimeValue	0..1	attr	The period between successive calls to Com_MainFunctionTx of the AUTOSAR COM module in seconds.
comEnableMDTForCyclicTransmission	Boolean	0..1	attr	Enables for the Com module of this EcuInstance the minimum delay time monitoring for cyclic and repeated transmissions (TransmissionModeTiming has cyclicTiming assigned or eventControlledTiming with numberOfRepetitions > 0).
commController	CommunicationController	1..*	aggr	CommunicationControllers of the ECU.
connector	CommunicationConnector	*	aggr	All channels controlled by a single controller.
diagnosticAddress	Integer	0..1	attr	An ECU specific ID for responses of diagnostic routines.
diagnosticProps	DiagnosticEcuProps	0..1	aggr	This represents the diagnostic-related properties of an entire ECU.
partition	EcuPartition	*	aggr	Optional definition of Partitions within an Ecu.
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.
pncPrepareSleepTimer	TimeValue	0..1	attr	Time in seconds the PNC state machine shall wait in PNC_PREPARE_SLEEP.

Attribute	Datatype	Mul.	Kind	Note
sleepModeSupported	Boolean	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.
wakeUpOverBusSupported	Boolean	1	attr	Driver support for wakeup over Bus.

Table F.24: EcuInstance

Class	EcucValueCollection			
Package	M2::AUTOSARTemplates::ECUCDescriptionTemplate			
Note	This represents the anchor point of the ECU configuration description. Tags: atp.recommendedPackage=EcucValueCollections			
Base	ARElement , ARObject , CollectableElement , Identifiable , MultilanguageReferrable , PackageableElement , Referrable			
Attribute	Datatype	Mul.	Kind	Note
ecuExtract	System	1	ref	Represents the extract of the System Configuration that is relevant for the ECU configured with that ECU Configuration Description.
ecucValue	EcucModuleConfigurationValues	1..*	ref	References to the configuration of individual software modules that are present on this ECU. atpVariation: [RS_ECUC_0079] Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime

Table F.25: EcucValueCollection

Class	EndToEndProtectionISignalPdu			
Package	M2::AUTOSARTemplates::SystemTemplate::EndToEndProtection			
Note	It is possible to protect the inter-ECU data exchange of safety-related ISignalGroups at the level of COM IPdus using protection mechanisms provided by E2E Library. For each ISignalGroup to be protected, a separate EndToEndProtectionISignalPdu element shall be created within the EndToEndProtectionSet. The EndToEndProtectionISignalPdu element refers to the ISignalGroup that is to be protected and to the ISignalPdu that transmits the protected ISignalGroup. The information how the referenced ISignalGroup shall be protected (through which E2E Profile and with which E2E settings) is defined in the EndToEndDescription element.			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
dataOffset	Integer	1	attr	This attribute defines the beginning offset (in bits) of the Array representation of the Signal Group (including CRC, counter and application signal group) in the IPdu. This attribute is mandatory and the dataOffset shall always be defined.
iSignalGroup	ISignalGroup	1	ref	Reference to the ISignalGroup that is to be protected.
iSignalPdu	ISignalPdu	1	ref	Reference to the ISignalPdu that transmits the protected ISignalGroup.

Table F.26: EndToEndProtectionISignalPdu

Class	EndToEndTransformationDescription			
Package	M2::AUTOSARTemplates::SystemTemplate::Transformer			
Note	EndToEndTransformationDescription holds these attributes which are profile specific and have the same value for all E2E transformers.			
Base	ARObject, Describable , TransformationDescription			
Attribute	Datatype	Mul.	Kind	Note
counterOffset	PositiveInteger	0..1	attr	Offset of the counter in the Data[] array in bits.
crcOffset	PositiveInteger	0..1	attr	Offset of the CRC in the Data[] array in bits.
dataIdMode	DataIdModeEnum	0..1	attr	This attribute describes the inclusion mode that is used to include the implicit two-byte Data ID in the one-byte CRC.
dataIdNibbleOffset	PositiveInteger	0..1	attr	Offset of the Data ID nibble in the Data[] array in bits.
maxDeltaCounter	PositiveInteger	0..1	attr	Maximum allowed gap between two counter values of two consecutive checks.
maxErrorStateInit	PositiveInteger	0..1	attr	Maximal number of checks in which ProfileStatus equal to E2E_P_ERROR was determined, within the last WindowSize checks, for the state E2E_SM_INIT.
maxErrorStateInvalid	PositiveInteger	0..1	attr	Maximal number of checks in which ProfileStatus equal to E2E_P_ERROR was determined, within the last WindowSize checks, for the state E2E_SM_INVALID.

Attribute	Datatype	Mul.	Kind	Note
maxErrorStateValid	PositiveInteger	0..1	attr	Maximal number of checks in which ProfileStatus equal to E2E_P_ERROR was determined, within the last WindowSize checks, for the state E2E_SM_VALID.
maxNoNewOrRepeatedData	PositiveInteger	0..1	attr	The maximum allowed amount of consecutive failed counter checks.
minOkStateInit	PositiveInteger	0..1	attr	Minimal number of checks in which ProfileStatus equal to E2E_P_OK was determined, within the last WindowSize checks, for the state E2E_SM_INIT.
minOkStateInvalid	PositiveInteger	0..1	attr	Minimal number of checks in which ProfileStatus equal to E2E_P_OK was determined, within the last WindowSize checks, for the state E2E_SM_INVALID.
minOkStateValid	PositiveInteger	0..1	attr	Minimal number of checks in which ProfileStatus equal to E2E_P_OK was determined, within the last WindowSize checks, for the state E2E_SM_VALID.
offset	PositiveInteger	0..1	attr	Offset of the E2E header in the Data[] array in bits.
profileBehavior	EndToEndProfileBehaviorEnum	0..1	attr	Behavior of the check functionality
profileName	NameToken	1	attr	Definition of the E2E profile.
syncCounterInit	PositiveInteger	0..1	attr	Number of checks required for validating the consistency of the counter that must be received with a valid counter (i.e. counter within the allowed lock-in range) after the detection of an unexpected behavior of a received counter.
upperHeaderBitsToShift	PositiveInteger	1	attr	This attribute describes the number of upper-header bits to be shifted. value = 0 or not present: shift of upper header is NOT performed. value > 0: the E2E Transformer on the protect-side, takes the first upperHeaderBitsToShift bits from the upper buffer (e.g. SOME/IP header part generated by SOME/IP transformer) and shifts them towards the lower bytes and bits within the Data[] for the length of the E2E header (e.g. 12 bytes in case of E2E Profile 4). This means the shift distance is fixed – it depends on the E2E header size – what is configured here is the number of bits that are to be shifted. This option is defined because the Some/IP header generated by SOME/IP transformer must be, due to compatibility between non-protected and E2E-protected communication, at the same position, which is before E2E header.
windowSize	PositiveInteger	0..1	attr	Size of the monitoring window for the E2E state machine.

Attribute	Datatype	Mul.	Kind	Note

Table F.27: EndToEndTransformationDescription

Class	ExecutableEntity (abstract)			
Package	M2::AUTOSARTemplates::CommonStructure::InternalBehavior			
Note	Abstraction of executable code.			
Base	ARObject, Identifiable , MultilanguageReferrable, Referrable			
Attribute	Datatype	Mul.	Kind	Note
activationReason	ExecutableEntityActivationReason	*	aggr	If the ExecutableEntity provides at least one activationReason element the RTE resp. BSW Scheduler shall provide means to read the activation vector of this executable entity execution. If no activationReason element is provided the feature of being able to determine the activating RTEEvent is disabled for this ExecutableEntity.
canEnterExclusiveArea	ExclusiveArea	*	ref	This means that the executable entity can enter/leave the referenced exclusive area through explicit API calls.
exclusiveAreaNestingOrder	ExclusiveAreaNestingOrder	*	ref	This represents the set of ExclusiveAreaNestingOrders recognized by this ExecutableEntity.
minimumStartInterval	TimeValue	1	attr	Specifies the time in seconds by which two consecutive starts of an ExecutableEntity are guaranteed to be separated.
reentrancyLevel	ReentrancyLevelEnum	0..1	attr	The reentrancy level of this ExecutableEntity. See the documentation of the enumeration type ReentrancyLevelEnum for details. Please note that nonReentrant interfaces can have also reentrant or multicoreReentrant implementations, and reentrant interfaces can also have multicoreReentrant implementations.
runsInsideExclusiveArea	ExclusiveArea	*	ref	The executable entity runs completely inside the referenced exclusive area.
swAddrMethod	SwAddrMethod	0..1	ref	Addressing method related to this code entity. Via an association to the same SwAddrMethod, it can be specified that several code entities (even of different modules or components) shall be located in the same memory without already specifying the memory section itself.

Table F.28: ExecutableEntity

Class	FlatInstanceDescriptor			
Package	M2::AUTOSARTemplates::CommonStructure::FlatMap			
Note	Represents exactly one node (e.g. a component instance or data element) of the instance tree of a software system. The purpose of this element is to map the various nested representations of this instance to a flat representation and assign a unique name (shortName) to it. Use cases: - Specify unique names of measurable data to be used by MCD tools - Specify unique names of calibration data to be used by MCD tool - Specify a unique name for an instance of a component prototype in the ECU extract of the system description Note that in addition it is possible to assign alias names via AliasNameAssignment.			
Base	ARObject, Identifiable , MultilanguageReferrable, Referrable			
Attribute	Datatype	Mul.	Kind	Note
ecuExtractReference	AtpFeature	0..1	iref	Refers to the instance in the ECU extract. This is valid only, if the FlatMap is used in the context of an ECU extract. The reference shall be such that it uniquely defines the object instance. For example, if a data prototype is declared as a role within an SwcInternalBehavior, it is not enough to state the SwcInternalBehavior as context and the aggregated data prototype as target. In addition, the reference shall also include the complete path identifying instance of the component prototype and the AtomicSoftwareComponentType, which is referred by the particular SwcInternalBehavior. Tags: xml.sequenceOffset=40
role	Identifier	0..1	ref	The role denotes the particular role of the downstream memory location described by this FlatInstanceDescriptor. It applies to use case where one upstream object results in multiple downstream objects, e.g. ModeDeclarationGroupPrototypes which are measurable. In this case the RTE will provide locations for current mode, previous mode and next mode.
swDataDefProps	SwDataDefProps	0..1	aggr	The properties of this FlatInstanceDescriptor.

Attribute	Datatype	Mul.	Kind	Note
upstreamReference	AtpFeature	0..1	iref	Refers to the instance in the context of an "upstream" descriptions, which could be the system or system extract description, the basic software module description or (if a flat map is used in preliminary context) a description of an atomic component or composition. This reference is optional in case the flat map is used in ECU context. The reference shall be such that it uniquely defines the object instance in the given context. For example, if a data prototype is declared as a role within an SwcInternalBehavior, it is not enough to state the SwcInternalBehavior as context and the aggregated data prototype as target. In addition, the reference shall also include the complete path identifying the instance of the component prototype that contains the particular instance of SwcInternalBehavior. Tags: xml.sequenceOffset=20

Table F.29: FlatInstanceDescriptor

Class	HwDescriptionEntity (abstract)			
Package	M2::AUTOSARTemplates::EcuResourceTemplate			
Note	This meta-class represents the ability to describe a hardware entity.			
Base	ARObject, Referrable			
Attribute	Datatype	Mul.	Kind	Note
hwAttributeValue	HwAttributeValue	*	aggr	This aggregation represents a particular hardware attribute value. Stereotypes: atpVariation Tags: vh.latestBindingTime=systemDesignTime xml.sequenceOffset=50
hwCategory	HwCategory	*	ref	One of the associations representing one particular category of the hardware entity. Tags: xml.sequenceOffset=30
hwType	HwType	0..1	ref	This association is used to assign an optional HwType which contains the common attribute values for all occurrences of this HwDescriptionEntity. Note that HwTypes can not be redefined and therefore shall not have a hwType reference.

Table F.30: HwDescriptionEntity

Class	HwElement			
Package	M2::AUTOSARTemplates::EcuResourceTemplate			
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			
Attribute	Datatype	Mul.	Kind	Note
hwElementConnection	HwElementConnector	*	aggr	This represents one particular connection between two hardware elements. Stereotypes: atpVariation Tags: 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: 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: vh.latestBindingTime=systemDesignTime xml.sequenceOffset=70

Table F.31: HwElement

Class	HwType			
Package	M2::AUTOSARTemplates::EcuResourceTemplate::HwElementCategory			
Note	This represents the ability to describe Hardware types on an abstract 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=HwTypes			
Base	ARElement , ARObject , CollectableElement , HwDescriptionEntity , Identifiable , MultilanguageReferrable , PackageableElement , Referrable			
Attribute	Datatype	Mul.	Kind	Note
—	—	—	—	—

Table F.32: HwType

Class	ISignal			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreCommunication			
Note	Signal of the Interaction Layer. The RTE supports a "signal fan-out" where the same System Signal is sent in different SignalIPdus to multiple receivers. To support the RTE "signal fan-out" each SignalIPdu contains ISignals. If the same System Signal is to be mapped into several SignalIPdus there is one ISignal needed for each ISignalToIPduMapping. ISignals describe the Interface between the Precompile configured RTE and the potentially Postbuild configured Com Stack (see ECUC Parameter Mapping). In case of the SystemSignalGroup an ISignal must be created for each SystemSignal contained in the SystemSignalGroup. Tags: atp.recommendedPackage=ISignals			
Base	ARObject,CollectableElement,FibexElement,Identifiable,Multilanguage Referrable,PackageableElement,Referrable			
Attribute	Datatype	Mul.	Kind	Note
dataTransformation	DataTransformation	0..1	ref	Optional reference to a DataTransformation which represents the transformer chain that is used to transform the data that shall be placed inside this ISignal. Stereotypes: atpSplittable; atpVariation Tags: atp.Splitkey=dataTransformation, variationPoint.shortLabel vh.latestBindingTime=codeGenerationTime
dataTypePolicy	DataTypePolicyEnum	1	attr	With the aggregation of SwDataDefProps an ISignal specifies how it is represented on the network. This representation follows a particular policy. Note that this causes some redundancy which is intended and can be used to support flexible development methodology as well as subsequent integrity checks. If the policy "networkRepresentationFromComSpec" is chosen the network representation from the ComSpec that is aggregated by the PortPrototype shall be used. If the "override" policy is chosen the requirements specified in the PortInterface and in the ComSpec are not fulfilled by the networkRepresentationProps. In case the System Description doesn't use a complete Software Component Description (VFB View) the "legacy" policy can be chosen.
iSignalProps	ISignalProps	0..1	aggr	Additional optional ISignal properties that may be stored in different files. Stereotypes: atpSplittable Tags: atp.Splitkey=iSignalProps

Attribute	Datatype	Mul.	Kind	Note
initValue	ValueSpecification	0..1	aggr	Optional definition of a ISignal's initValue in case the System Description doesn't use a complete Software Component Description (VFB View). This supports the inclusion of legacy system signals. This value can be used to configure the Signal's "InitValue". If a full DataMapping exist for the SystemSignal this information may be available from a configured SenderComSpec and ReceiverComSpec. In this case the initvalues in SenderComSpec and/or ReceiverComSpec override this optional value specification. Further restrictions apply from the RTE specification.
length	Integer	1	attr	Size of the signal in bits. The size needs to be derived from the mapped VariableDataPrototype according to the mapping of primitive DataTypes to BaseTypes as used in the RTE. Indicates maximum size for dynamic length signals. The ISignal length of zero bits is allowed.
networkRepresentation	SwDataDefProps	0..1	aggr	Specification of the actual network representation. The usage of SwDataDefProps for this purpose is restricted to the attributes compuMethod and baseType. The optional baseType attributes "memAlignment" and "byteOrder" shall not be used. The attribute "dataTypePolicy" in the SystemTemplate element defines whether this network representation shall be ignored and the information shall be taken over from the network representation of the ComSpec. If "override" is chosen by the system integrator the network representation can violate against the requirements defined in the PortInterface and in the network representation of the ComSpec. In case that the System Description doesn't use a complete Software Component Description (VFB View) this element is used to configure "ComSignalDataInvalidValue" and the Data Semantics.
systemSignal	SystemSignal	1	ref	Reference to the System Signal that is supposed to be transmitted in the ISignal.

Attribute	Datatype	Mul.	Kind	Note
transformationISignalProps	TransformationISignalProps	*	aggr	A transformer chain consists of an ordered list of transformers. The ISignal specific configuration properties for each transformer are defined in the TransformationISignalProps class. The transformer configuration properties that are common for all ISignals are described in the TransformationTechnology class.

Table F.33: ISignal

Class	ISignalGroup			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreCommunication			
Note	SignalGroup of the Interaction Layer. The RTE supports a "signal fan-out" where the same System Signal Group is sent in different SignalIPdus to multiple receivers. An ISignalGroup refers to a set of ISignals that shall always be kept together. A ISignalGroup represents a COM Signal Group. Therefore it is recommended to put the ISignalGroup in the same Package as ISignals (see atp.recommendedPackage) Tags: atp.recommendedPackage=ISignalGroup			
Base	ARObject,CollectableElement,FibexElement,Identifiable,MultilanguageReferrable,PackageableElement,Referrable			
Attribute	Datatype	Mul.	Kind	Note
comBasedSignalGroupTransformation	DataTransformation	0..1	ref	Optional reference to a DataTransformation which represents the transformer chain that is used to transform the data that shall be placed inside this ISignalGroup based on the COMBasedTransformer approach. Stereotypes: atpSplittable; atpVariation Tags: atp.Splitkey=comBasedSignalGroupTransformation, variationPoint.shortLabel vh.latestBindingTime=codeGenerationTime
iSignal	ISignal	*	ref	Reference to a set of ISignals that shall always be kept together.
systemSignalGroup	SystemSignalGroup	1	ref	Reference to the SystemSignalGroup that is defined on VFB level and that is supposed to be transmitted in the ISignalGroup.
transformationISignalProps	TransformationISignalProps	*	aggr	A transformer chain consists of an ordered list of transformers. The ISignalGroup specific configuration properties for each transformer are defined in the TransformationISignalProps class. The transformer configuration properties that are common for all ISignalGroups are described in the TransformationTechnology class.

Table F.34: ISignalGroup

Class	ISignalIPdu			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreCommunication			
Note	Represents the IPdus handled by Com. The ISignalIPdu assembled and disassembled in AUTOSAR COM consists of one or more signals. In case no multiplexing is performed this IPdu is routed to/from the Interface Layer. A maximum of one dynamic length signal per IPdu is allowed. Tags: atp.recommendedPackage=Pdus			
Base	ARObject,CollectableElement,FibexElement,IPdu,Identifiable,Multilanguage Referrable,PackageableElement,Pdu,Referrable			
Attribute	Datatype	Mul.	Kind	Note
iPduTiming Specification	IPduTiming	0..1	aggr	Timing specification for Com IPdus (Transmission Modes). This information is mandatory for the sender in a System Extract. This information may be omitted on receivers in a System Extract. atpVariation: The timing of a Pdu can vary. Stereotypes: atpVariation Tags: vh.latestBindingTime=postBuild
iSignalToPduMapping	ISignalToIPduMapping	*	aggr	Definition of SignalToIPduMappings included in the SignalIPdu. atpVariation: The content of a PDU can be variable. Stereotypes: atpVariation Tags: vh.latestBindingTime=postBuild
pduCounter	SignalIPduCounter	0..1	aggr	An included Pdu counter is used to ensure that a sequence of Pdus is maintained. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime
pduReplication	SignalIPduReplication	0..1	aggr	Pdu Replication is a form of redundancy where the data content of one ISignalIPdu (source) is transmitted inside a set of replica ISignalIPdus. These ISignalIPdus (copies) have different Pdu IDs, identical PduCounters, identical data content and are transmitted with the same frequency. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime
unusedBitPattern	Integer	1	attr	AUTOSAR COM and AUTOSAR IPDUM are filling not used areas of an IPDU with this bit-pattern. This attribute is mandatory to avoid undefined behavior. This byte-pattern will be repeated throughout the IPdu.

Table F.35: ISignalIPdu

Class	Identifiable (abstract)			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::Identifiable			
Note	Instances of this class can be referred to by their identifier (within the namespace borders). In addition to this, Identifiables are objects which contribute significantly to the overall structure of an AUTOSAR description. In particular, Identifiables might contain Identifiables.			
Base	ARObject, MultilanguageReferrable, Referrable			
Attribute	Datatype	Mul.	Kind	Note
desc	MultiLanguageOverviewParagraph	0..1	aggr	This represents a general but brief (one paragraph) description what the object in question is about. It is only one paragraph! Desc is intended to be collected into overview tables. This property helps a human reader to identify the object in question. More elaborate documentation, (in particular how the object is built or used) should go to "introduction". Tags: xml.sequenceOffset=-60
category	CategoryString	0..1	attr	The category is a keyword that specializes the semantics of the Identifiable. It affects the expected existence of attributes and the applicability of constraints. Tags: xml.sequenceOffset=-50
adminData	AdminData	0..1	aggr	This represents the administrative data for the identifiable object. Tags: xml.sequenceOffset=-40
annotation	Annotation	*	aggr	Possibility to provide additional notes while defining a model element (e.g. the ECU Configuration Parameter Values). These are not intended as documentation but are mere design notes. Tags: xml.sequenceOffset=-25
introduction	DocumentationBlock	0..1	aggr	This represents more information about how the object in question is built or is used. Therefore it is a DocumentationBlock. Tags: xml.sequenceOffset=-30

Attribute	Datatype	Mul.	Kind	Note
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 F.36: Identifiable

Primitive	Identifier			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::Primitive Types			
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	Datatype	Mul.	Kind	Note
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 F.37: Identifier

Class	Implementation (abstract)			
Package	M2::AUTOSARTemplates::CommonStructure::Implementation			
Note	Description of an implementation a single software component or module.			
Base	ARElement, ARObject, CollectableElement, Identifiable, Multilanguage Referrable, PackageableElement, Referrable			
Attribute	Datatype	Mul.	Kind	Note
buildActionManifest	BuildActionManifest	0..1	ref	A manifest specifying the intended build actions for the software delivered with this implementation. Stereotypes: atpVariation Tags: vh.latestBindingTime=codeGenerationTime
codeDescriptor	Code	1..*	aggr	Specifies the provided implementation code.
compiler	Compiler	*	aggr	Specifies the compiler for which this implementation has been released
generatedArtifact	DependencyOnArtifact	*	aggr	Relates to an artifact that will be generated during the integration of this Implementation by an associated generator tool. Note that this is an optional information since it might not always be in the scope of a single module or component to provide this information. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime
hwElement	HwElement	*	ref	The hardware elements (e.g. the processor) required for this implementation.
linker	Linker	*	aggr	Specifies the linker for which this implementation has been released.
mcSupport	McSupportData	0..1	aggr	The measurement & calibration support data belonging to this implementation. The aggregation is «atpSplitable» because in case of an already existing BSW Implementation model, this description will be added later in the process, namely at code generation time. Stereotypes: atpSplitable Tags: atp.Splitkey=mcSupport
programmingLanguage	ProgrammingLanguageEnum	1	attr	Programming language the implementation was created in.
requiredArtifact	DependencyOnArtifact	*	aggr	Specifies that this Implementation depends on the existence of another artifact (e.g. a library). This aggregation of DependencyOnArtifact is subject to variability with the purpose to support variability in the implementations. Different algorithms in the implementation might cause different dependencies, e.g. the number of used libraries. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime

Attribute	Datatype	Mul.	Kind	Note
requiredGeneratorTool	DependencyOnArtifact	*	aggr	Relates this Implementation to a generator tool in order to generate additional artifacts during integration. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime
resourceConsumption	ResourceConsumption	1	aggr	All static and dynamic resources for each implementation are described within the ResourceConsumption class. Stereotypes: atpSplittable Tags: atp.Splitkey=shortName
swVersion	RevisionLabelString	1	attr	Software version of this implementation. The numbering contains three levels (like major, minor, patch), its values are vendor specific.
swcBswMapping	SwcBswMapping	0..1	ref	This allows a mapping between an SWC and a BSW behavior to be attached to an implementation description (for AUTOSAR Service, ECU Abstraction and Complex Driver Components). It is up to the methodology to define whether this reference has to be set for the Swc- or BswImplementation or for both.
usedCodeGenerator	String	0..1	attr	Optional: code generator used.
vendorId	PositiveInteger	1	attr	Vendor ID of this Implementation according to the AUTOSAR vendor list

Table F.38: Implementation

Class	InstantiationTimingEventProps			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Composition			
Note	This meta class represents the ability to refine a timing event for particular instances of a software component. This supports then an instance specific timing.			
Base	ARObject, InstantiationRTEEventProps			
Attribute	Datatype	Mul.	Kind	Note
period	TimeValue	1	attr	

Table F.39: InstantiationTimingEventProps

Class	InternalBehavior (abstract)			
Package	M2::AUTOSARTemplates::CommonStructure::InternalBehavior			
Note	Common base class (abstract) for the internal behavior of both software components and basic software modules/clusters.			
Base	ARObject, AtpClassifier, AtpFeature, AtpStructureElement, Identifiable , Multilanguage Referrable, Referrable			
Attribute	Datatype	Mul.	Kind	Note

Attribute	Datatype	Mul.	Kind	Note
constantMemory	ParameterDataPrototype	*	aggr	Describes a read only memory object containing characteristic value(s) implemented by this InternalBehavior. The shortName of ParameterDataPrototype has to be equal to the 'C' identifier of the described constant. The characteristic value(s) might be shared between SwComponentPrototypes of the same SwComponentType. The aggregation of constantMemory is subject to variability with the purpose to support variability in the software component or module implementations. Typically different algorithms in the implementation are requiring different number of memory objects. Stereotypes: atpSplitable; atpVariation Tags: atp.Splitkey=shortName, variation Point.shortLabel vh.latestBindingTime=preCompileTime
constantValueMapping	ConstantSpecificationMappingSet	*	ref	Reference to the ConstantSpecificationMapping to be applied for the particular InternalBehavior Stereotypes: atpSplitable Tags: atp.Splitkey=constantValueMapping
dataTypeMapping	DataTypeMappingSet	*	ref	Reference to the DataTypeMapping to be applied for the particular InternalBehavior Stereotypes: atpSplitable Tags: atp.Splitkey=dataTypeMapping
exclusiveArea	ExclusiveArea	*	aggr	This specifies an ExclusiveArea for this InternalBehavior. The exclusiveArea is local to the component resp. module. The aggregation of ExclusiveAreas is subject to variability. Note: the number of ExclusiveAreas might vary due to the conditional existence of RunnableEntities or BswModuleEntities. Stereotypes: atpSplitable; atpVariation Tags: atp.Splitkey=shortName, variation Point.shortLabel vh.latestBindingTime=preCompileTime
exclusiveAreaNestingOrder	ExclusiveAreaNestingOrder	*	aggr	This represents the set of ExclusiveAreaNestingOrder owned by the InternalBehavior. Stereotypes: atpSplitable; atpVariation Tags: atp.Splitkey=shortName, variation Point.shortLabel vh.latestBindingTime=preCompileTime

Attribute	Datatype	Mul.	Kind	Note
staticMemory	VariableDataPrototype	*	aggr	Describes a read and writeable static memory object representing measurement variables implemented by this software component. Static is used in the meaning of non temporary and does not necessarily specify a linker encapsulation. This kind of memory is only supported if supportsMultipleInstantiation is FALSE. The shortName of the VariableDataPrototype has to be equal with the 'C' identifier of the described variable. The aggregation of staticMemory is subject to variability with the purpose to support variability in the software components implementations. Typically different algorithms in the implementation are requiring different number of memory objects. Stereotypes: atpSplittable; atpVariation Tags: atp.Splitkey=shortName, variation Point.shortLabel vh.latestBindingTime=preCompileTime

Table F.40: InternalBehavior

Class	McDataInstance			
Package	M2::AUTOSARTemplates::CommonStructure::MeasurementCalibrationSupport			
Note	Describes the specific properties of one data instance in order to support measurement and/or calibration of this data instance. The most important attributes are: • Its shortName is copied from the ECU Flat map (if applicable) and will be used as identifier and for display by the MC system. • The category is copied from the corresponding data type (ApplicationDataType if defined, otherwise ImplementationDataType) as far as applicable. • The symbol is the one used in the programming language. It will be used to find out the actual memory address by the final generation tool with the help of linker generated information. It is assumed that in the M1 model this part and all the aggregated and referred elements (with the exception of the Flat Map and the references from ImplementationElementInParameterInstanceRef and McAccessDetails) are completely generated from "upstream" information. This means, that even if an element like e.g. a CompuMethod is only used via reference here, it will be copied into the M1 artifact which holds the complete McSupportData for a given Implementation.			
Base	ARObject, Identifiable, MultilanguageReferrable, Referrable			
Attribute	Datatype	Mul.	Kind	Note
arraySize	PositiveInteger	0..1	attr	The existence of this attribute turns the data instance into an array of data. The attribute determines the size of the array in terms of number of elements.

Attribute	Datatype	Mul.	Kind	Note
displayIdentifier	McIdIdentifier	0..1	attr	An optional attribute to be used to set the ASAM ASAP2 DISPLAY_IDENTIFIER attribute.
flatMapEntry	FlatInstanceDescriptor	0..1	ref	Reference to the corresponding entry in the ECU Flat Map. This allows to trace back to the original specification of the generated data instance. This link shall be added by the RTE generator mainly for documentation purposes. The reference is optional because • The McDataInstance may represent an array or struct in which only the subElements correspond to FlatMap entries. • The McDataInstance may represent a task local buffer for rapid prototyping access which is different from the "main instance" used for measurement access.
instanceInMemory	ImplementationElementInParameterInstanceRef	0..1	aggr	Reference to the corresponding data instance in the description of calibration data structures published by the RTE generator. This is used to support emulation methods inside the ECU, it is not required for A2L generation.
mcDataAccessDetails	McDataAccessDetails	0..1	aggr	Refers to "upstream" information on how the RTE uses this data instance. Use Case: Rapid Prototyping
mcDataAssignment	RoleBasedMcDataAssignment	*	aggr	An assignment between McDataInstances.
resultingProperties	SwDataDefProperties	0..1	aggr	These are the generated properties resulting from decisions taken by the RTE generator for the actually implemented data instance. Only those properties are relevant here, which are needed for the measurement and calibration system.
role	Identifier	0..1	ref	An optional attribute to be used for additional information on the role of this data instance, for example in the context of rapid prototyping.
subElement	McDataInstance	*	aggr	This relation indicates, that the target element is part of a "struct" which is given by the source element. This information will be used by the final generator to set up the correct addressing scheme. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime

Attribute	Datatype	Mul.	Kind	Note
symbol	SymbolString	0..1	ref	This String is used to determine the memory address during final generation of the MC configuration data (e.g. "A2L" file) . It shall be the name of the element in the programming language such that it can be identified in linker generated information. In case the McDataInstance is part of composite data in the programming language, the symbol String may include parts denoting the element context, unless the context is given by the symbol attribute of an enclosing McDataInstance. This means in particular for the C language that the "." character shall be used as a separator between the name of a "struct" variable the name of one of its elements. The symbol can differ from the shortName in case of generated C data declarations. It is an optional attribute since it may be missing in case the instance represents an element (e.g. a single array element) which has no name in the linker map.

Table F.41: McDataInstance

Class	McSupportData			
Package	M2::AUTOSARTemplates::CommonStructure::MeasurementCalibrationSupport			
Note	Root element for all measurement and calibration support data related to one Implementation artifact on an ECU. There shall be one such element related to the RTE implementation (if it owns MC data) and a separate one for each module or component, which owns private MC data.			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
emulationSupport	McSwEmulationMethodSupport	*	aggr	Describes the calibration method used by the RTE. This information is not needed for A2L generation, but to setup software emulation in the ECU. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime
mcParameterInstance	McDataInstance	*	aggr	A data instance to be used for calibration. Stereotypes: atpSplittable; atpVariation Tags: atp.Splitkey=shortName, variation Point.shortLabel vh.latestBindingTime=postBuild

Attribute	Datatype	Mul.	Kind	Note
mcVariableInstance	McDataInstance	*	aggr	A data instance to be used for measurement. Stereotypes: atpSplitable; atpVariation Tags: atp.Splitkey=shortName, variation Point.shortLabel vh.latestBindingTime=postBuild
measurableSystemConstantValues	SwSystemconstantValueSet	*	ref	Sets of system constant values to be transferred to the MCD system, because the system constants have been specified with "swCalibrationAccess" = readonly.

Table F.42: McSupportData

Class	MemorySection			
Package	M2::AUTOSARTemplates::CommonStructure::ResourceConsumption::MemorySectionUsage			
Note	Provides a description of an abstract memory section used in the Implementation for code or data. It shall be declared by the Implementation Description of the module or component, which actually allocates the memory in its code. This means in case of data prototypes which are allocated by the RTE, that the generated Implementation Description of the RTE shall contain the corresponding MemorySections. The attribute "symbol" (if symbol is missing: "shortName") defines the module or component specific section name used in the code. For details see the document "Specification of Memory Mapping". Typically the section name is build according the pattern: <SwAddrMethod shortName>[_<further specialization nominator>][_<alignment>] where • [<SwAddrMethod shortName>] is the shortName of the referenced SwAddrMethod • [<further specialization nominator>] is an optional infix to indicate the specialization in the case that several MemorySections for different purpose of the same Implementation Description referring to the same or equally named SwAddrMethods. • [<alignment>] is the alignment attributes value and is only applicable in the case that the memoryAllocationKeywordPolicy value of the referenced SwAddrMethod is set to addrMethodShortNameAndAlignment MemorySection used to Implement the code of RunnableEntitys and BswSchedulableEntitys shall have a symbol (if missing: shortName) identical to the referred SwAddrMethod to conform to the generated RTE header files. In addition to the section name described above, a prefix is used in the corresponding macro code in order to define a name space. This prefix is by default given by the shortName of the BswModuleDescription resp. the SwComponentType. It can be superseded by the prefix attribute.			
Base	ARObject,Identifiable,MultilanguageReferrable,Referrable			
Attribute	Datatype	Mul.	Kind	Note

Attribute	Datatype	Mul.	Kind	Note
alignment	AlignmentType	0..1	attr	The attribute describes the alignment of objects within this memory section.
executable Entity	ExecutableEntity	*	ref	Reference to the ExecutableEntities located in this section. This allows to locate different ExecutableEntities in different sections even if the associated SwAddrmethod is the same. This is applicable to code sections only.
memClass Symbol	CIdentifier	0..1	ref	Defines a specific symbol in order to generate the compiler abstraction "memclass" code for this MemorySection. The existence of this attribute supersedes the usage of swAddrmethod.shortName for this purpose. The complete name of the "memclass" preprocessor symbol is constructed as <prefix>_<memClassSymbol> where prefix is defined in the same way as for the enclosing MemorySection. See also AUTOSAR_SWS_CompilerAbstraction SWS_COMPILER_00040.
option	Identifier	*	ref	This attribute introduces the ability to specify further intended properties of this MemorySection. The following two values are standardized (to be used for code sections only and exclusively to each other): • INLINE - The code section is declared with the compiler abstraction macro INLINE. • LOCAL_INLINE - The code section is declared with the compiler abstraction macro LOCAL_INLINE In both cases (INLINE and LOCAL_INLINE) the inline expansion depends on the compiler specific implementation of these macros. Depending on this, the code section either corresponds to an actual section in memory or is put into the section of the caller. See AUTOSAR_SWS_CompilerAbstraction for more details.
prefix	SectionNamePrefix	0..1	ref	The prefix used to set the memory section's namespace in the code. The existence of a prefix element supersedes rules for a default prefix (such as the BswModuleDescription's shortName). This allows the user to define several name spaces for memory sections within the scope of one module, cluster or SWC.
size	PositiveInteger	0..1	attr	The size in bytes of the section.

Attribute	Datatype	Mul.	Kind	Note
swAddrmethod	SwAddrMethod	1	ref	This association indicates that this module specific (abstract) memory section is part of an overall SwAddrMethod, referred by the upstream declarations (e.g. calibration parameters, data element prototypes, code entities) which share a common addressing strategy. This can be evaluated for the ECU configuration of the build support. This association shall always be declared by the Implementation description of the module or component, which allocates the memory in its code. This means in case of data prototypes which are allocated by the RTE, that the software components only declare the grouping of its data prototypes to SwAddrMethods, and the generated Implementation Description of the RTE actually sets up this association.
symbol	Identifier	0..1	ref	Defines the section name as explained in the main description. By using this attribute for code generation (instead of the shortName) it is possible to define several different MemorySections having the same name - e.g. symbol = CODE - but using different sectionNamePrefixes.

Table F.43: MemorySection

Enumeration	ModeActivationKind
Package	M2::AUTOSARTemplates::CommonStructure::ModeDeclaration
Note	Kind of mode switch condition used for activation of an event, as further described for each enumeration field.
Literal	Description
onEntry	On entering the referred mode.
onExit	On exiting the referred mode.
onTransition	On transition of the 1st referred mode to the 2nd referred mode.

Table F.44: ModeActivationKind

Class	ParameterInAtomicSWCTypeInstanceRef			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::Data Elements::InstanceRefsUsage			
Note	This class implements an instance reference which can be applied for variables as well as for parameters.			
Base	ARObject, AtpInstanceRef			
Attribute	Datatype	Mul.	Kind	Note
base	AtomicSwComp onentType	1	ref	Stereotypes: atpDerived Tags: xml.sequence Offset=10

Attribute	Datatype	Mul.	Kind	Note
contextDataPrototype (ordered)	ApplicationCompositeElementDataPrototype	*	ref	This is the context in a compositeDataType. Tags: xml.sequenceOffset=40
portPrototype	PortPrototype	0..1	ref	This is the port providing the variable or the entry point to the variable structure. Tags: xml.sequenceOffset=20
rootParameterDataPrototype	DataPrototype	0..1	ref	This represents the entry point for references into a CompositeDataType. Tags: xml.sequenceOffset=30
targetDataPrototype	DataPrototype	1	ref	This is the target of the instance ref Tags: xml.sequenceOffset=50

Table F.45: ParameterInAtomicSWCTypeInstanceRef

Class	PostBuildVariantCriterionValueSet			
Package	M2::AUTOSARTemplates::GenericStructure::VariantHandling			
Note	This meta-class represents the ability to denote one set of postBuildVariantCriterionValues. Tags: atp.recommendedPackage=PostBuildVariantCriterionValueSets			
Base	ARElement , ARObject , CollectableElement , Identifiable , MultilanguageReferrable , PackageableElement , Referrable			
Attribute	Datatype	Mul.	Kind	Note
postBuildVariantCriterionValue	PostBuildVariantCriterionValue	*	aggr	This is one particular postbuild variant criterion/value pair being part of the PostBuildVariantSet.

Table F.46: PostBuildVariantCriterionValueSet

Primitive	Ref			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::PrimitiveTypes			
Note	This primitive denotes a name based reference. For detailed syntax see the xsd.pattern. • first slash (relative or absolute reference) [optional] • Identifier [required] • a sequence of slashes and Identifiers [optional] This primitive is used by the meta-model tools to create the references. Tags: xml.xsd.customType=REF; xml.xsd.pattern=/?[a-zA-Z][a-zA-Z0-9_]{0,127}/([a-zA-Z][a-zA-Z0-9_]{0,127})*; xml.xsd.type=string			
Attribute	Datatype	Mul.	Kind	Note

Attribute	Datatype	Mul.	Kind	Note
base	Identifier	0..1	ref	This attribute reflects the base to be used for this reference. Tags: xml.attribute=true
index	PositiveInteger	0..1	attr	This attribute supports the use case to point on specific elements in an array. This is in particular required if arrays are used to implement particular data objects. Tags: xml.attribute=true

Table F.47: Ref

[constr_2552] Index attribute is only valid for arrays [The index attribute in references is valid only if the reference target is an ApplicationArrayElement or if the reference target is an ImplementationDataTypeElement owned by an Implementation-DataType/ImplementationDataTypeElement of category ARRAY and has an attribute maxNumberOfElements/arraySize.]()

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

Table F.48: Referrable

Class	RootSwCompositionPrototype			
Package	M2::AUTOSARTemplates::SystemTemplate			
Note	The RootSwCompositionPrototype represents the top-level-composition of software components within a given System. According to the use case of the System, this may for example be the a more or less complete VFB description, the software of a System Extract or the software of a flat ECU Extract with only atomic SWCs. Therefore the RootSwComposition will only occasionally contain all atomic software components that are used in a complete VFB System. The OEM is primarily interested in the required functionality and the interfaces defining the integration of the Software Component into the System. The internal structure of such a component contains often substantial intellectual property of a supplier. Therefore a top-level software composition will often contain empty compositions which represent subsystems. The contained SwComponentPrototypes are fully specified by their SwComponentTypes (including PortPrototypes, PortInterfaces, VariableDataPrototypes, SwcInternalBehavior etc.), and their ports are interconnected using SwConnectorPrototypes.			
Base	ARObject, AtpFeature, AtpPrototype, Identifiable , MultilanguageReferrable, Referrable			
Attribute	Datatype	Mul.	Kind	Note
calibrationParameterValueSet	CalibrationParameterValueSet	*	ref	Used CalibrationParameterValueSet for instance specific initialization of calibration parameters. Stereotypes: atpSplitable Tags: atp.Splitkey=calibrationParameterValueSet
flatMap	FlatMap	0..1	ref	The FlatMap used in the scope of this RootSwCompositionPrototype. Stereotypes: atpSplitable Tags: atp.Splitkey=flatMap
softwareComposition	CompositionSwComponentType	1	tref	We assume that there is exactly one top-level composition that includes all Component instances of the system Stereotypes: isOfType

Table F.49: RootSwCompositionPrototype

Class	Sdg			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::SpecialData			
Note	Sdg (SpecialDataGroup) is a generic model which can be used to keep arbitrary information which is not explicitly modeled in the meta-model. Sdg can have various contents as defined by sdgContentsType. Special Data should only be used moderately since all elements should be defined in the meta-model. Thereby SDG should be considered as a temporary solution when no explicit model is available. If an sdgCaption is available, it is possible to establish a reference to the sdg structure.			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note

Attribute	Datatype	Mul.	Kind	Note
gid	NameToken	1	attr	This attributes specifies an identifier. Gid comes from the SGML/XML-Term "Generic Identifier" which is the element name in XML. The role of this attribute is the same as the name of an XML - element. Tags: xml.attribute=true
sdgCaption	SdgCaption	0..1	aggr	This aggregation allows to assign the properties of Identifiable to the sdg. By this, a shortName etc. can be assigned to the Sdg. Tags: xml.sequenceOffset=20
sdgCaptionRef	SdgCaption	0..1	ref	This association allows to reuse an already existing caption. Tags: xml.name=SDG-CAPTION-REF; xml.sequenceOffset=25
sdgContentsType	SdgContents	0..1	aggr	This is the content of the Sdg. Tags: xml.roleElement=false; xml.roleWrapperElement=false; xml.sequenceOffset=30; xml.typeElement=false; xml.typeWrapperElement=false

Table F.50: Sdg

Class	SenderReceiverToSignalMapping			
Package	M2::AUTOSARTemplates::SystemTemplate::DataMapping			
Note	Mapping of a sender receiver communication data element with a primitive datatype to a signal.			
Base	ARObject, DataMapping			
Attribute	Datatype	Mul.	Kind	Note
dataElement	VariableDataPrototype	1	iref	Reference to the data element, which ought to be sent over the Communication bus.
systemSignal	SystemSignal	1	ref	Reference to the system signal used to carry the data element.

Table F.51: SenderReceiverToSignalMapping

Primitive	String
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::PrimitiveTypes
Note	This represents a String in which white-space must be normalized before processing. For example: in order to compare two Strings: • leading and trailing white-space needs to be removed • consecutive white-space (blank, cr, lf, tab) needs to be replaced by one blank. Tags: xml.xsd.customType=STRING; xml.xsd.type=string

Table F.52: String

Class	SwServiceArg			
Package	M2::AUTOSARTemplates::CommonStructure::ServiceProcessTask			
Note	Specifies the properties of a data object exchanged during the call of an SwService, e.g. an argument or a return value. The SwServiceArg can also be used in the argument list of a C-macro. For this purpose the category shall be set to "MACRO". A reference to implementationDataType can optional be added if the actual argument has an implementationDataType.			
Base	ARObject, Identifiable , MultilanguageReferrable, Referrable			
Attribute	Datatype	Mul.	Kind	Note
direction	ArgumentDirectionEnum	0..1	attr	Specifies the direction of the data transfer. The direction shall indicate the direction of the actual information that is being consumed by the caller and/or the callee, not the direction of formal arguments in C. The attribute is optional for backwards compatibility reasons. For example, if a pointer is used to pass a memory address for the expected result, the direction shall be "out". If a pointer is used to pass a memory address with content to be read by the callee, its direction shall be "in". Tags: xml.sequenceOffset=10
swArraysize	ValueList	0..1	aggr	This turns the argument of the service to an array. Tags: xml.sequenceOffset=20
swDataDefProps	SwDataDefProps	0..1	aggr	Data properties of this SwServiceArg. Tags: xml.sequenceOffset=30

Table F.53: SwServiceArg

Class	SwSystemconst			
Package	M2::AUTOSARTemplates::CommonStructure::SystemConstant			
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			
Attribute	Datatype	Mul.	Kind	Note

Attribute	Datatype	Mul.	Kind	Note
swDataDefProps	SwDataDefProps	0..1	aggr	This denotes the data definition properties of the system constant. In particular it is the limits and - in case the system constant is an enumeration - the compu method. Tags: xml.sequenceOffset=40

Table F.54: SwSystemconst

Class	«atpMixedString» SwSystemconstDependentFormula (abstract)			
Package	M2::AUTOSARTemplates::GenericStructure::VariantHandling			
Note	This class represents an expression depending on system constants.			
Base	ARObject,FormulaExpression			
Attribute	Datatype	Mul.	Kind	Note
sysc	SwSystemconst	1	ref	This refers to a system constant. The internal (coded) value of the system constant shall be used. Tags: xml.sequenceOffset=50
syscString	SwSystemconst	1	ref	syscString indicates that the referenced system constant shall be evaluated as a string according to [TPS_SWCT_01431] .

Table F.55: SwSystemconstDependentFormula

Class	SwSystemconstantValueSet			
Package	M2::AUTOSARTemplates::GenericStructure::VariantHandling			
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,PackageableElement, Referrable			
Attribute	Datatype	Mul.	Kind	Note
swSystemconstantValue	SwSystemconst Value	*	aggr	This is one particular value of a system constant.

Table F.56: SwSystemconstantValueSet

Class	SwcBswMapping			
Package	M2::AUTOSARTemplates::CommonStructure::SwcBswMapping			
Note	Maps an SwcInternalBehavior to an BswInternalBehavior. This is required to coordinate the API generation and the scheduling for AUTOSAR Service Components, ECU Abstraction Components and Complex Driver Components by the RTE and the BSW scheduling mechanisms. Tags: atp.recommendedPackage=SwcBswMappings			
Base	ARElement , ARObject , AtpClassifier , AtpFeature , AtpStructureElement , CollectableElement , Identifiable , MultilanguageReferrable , PackageableElement , Referrable			
Attribute	Datatype	Mul.	Kind	Note
bswBehavior	BswInternalBehavior	1	ref	The mapped BswInternalBehavior
runnableMapping	SwcBswRunnableMapping	*	aggr	A mapping between a pair of SWC and BSW runnables. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime
swcBehavior	SwcInternalBehavior	1	ref	The mapped SwcInternalBehavior.
synchronizedModeGroup	SwcBswSynchronizedModeGroupPrototype	*	aggr	A pair of SWC and BSW mode group prototypes to be synchronized by the scheduler. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime
synchronizedTrigger	SwcBswSynchronizedTrigger	*	aggr	A pair of SWC and BSW Triggers to be synchronized by the scheduler. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime

Table F.57: SwcBswMapping

Class	System			
Package	M2::AUTOSARTemplates::SystemTemplate			
Note	The top level element of the System Description. The System description defines five major elements: Topology, Software, Communication, Mapping and Mapping Constraints. The System element directly aggregates the elements describing the Software, Mapping and Mapping Constraints; it contains a reference to an ASAM FIBEX description specifying Communication and Topology. Tags: atp.recommendedPackage=Systems			
Base	ARElement , ARObject , AtpClassifier , AtpFeature , AtpStructureElement , CollectableElement , Identifiable , MultilanguageReferrable , PackageableElement , Referrable			
Attribute	Datatype	Mul.	Kind	Note
clientIdDefinitionSet	ClientIdDefinitionSet	*	ref	Set of Client Identifiers that are used for inter-ECU client-server communication in the System.
containerIPduHeaderByteOrder	ByteOrderEnum	0..1	attr	Defines the byteOrder of the header in ContainerIPdus.

Attribute	Datatype	Mul.	Kind	Note
ecuExtractVersion	RevisionLabelString	0..1	attr	Version number of the Ecu Extract.
fibexElement	FibexElement	*	ref	Reference to ASAM FIBEX elements specifying Communication and Topology. All Fibex Elements used within a System Description shall be referenced from the System Element. atpVariation: In order to describe a product-line, all FibexElements can be optional. Stereotypes: atpVariation Tags: vh.latestBindingTime=postBuild
mapping	SystemMapping	*	aggr	Aggregation of all mapping aspects (mapping of SW components to ECUs, mapping of data elements to signals, and mapping constraints). In order to support OEM / Tier 1 interaction and shared development for one common System this aggregation is atpSplittable and atpVariation. The content of SystemMapping can be provided by several parties using different names for the SystemMapping. This element is not required when the System description is used for a network-only use-case. Stereotypes: atpSplittable; atpVariation Tags: atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=postBuild
pncVectorLength	PositiveInteger	0..1	attr	Length of the partial networking request release information vector (in bytes).
pncVectorOffset	PositiveInteger	0..1	attr	Absolute offset (with respect to the NM-PDU) of the partial networking request release information vector that is defined in bytes as an index starting with 0.
rootSoftwareComposition	RootSwCompositionPrototype	0..1	aggr	Aggregation of the root software composition, containing all software components in the System in a hierarchical structure. This element is not required when the System description is used for a network-only use-case. atpVariation: The RootSwCompositionPrototype can vary. Stereotypes: atpSplittable; atpVariation Tags: atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=systemDesignTime

Attribute	Datatype	Mul.	Kind	Note
systemDocumentation	Chapter	*	aggr	Possibility to provide additional documentation while defining the System. The System documentation can be composed of several chapters. Stereotypes: atpSplitable; atpVariation Tags: atp.Splitkey=shortName, variationPoint.shortLabel vh.latestBindingTime=systemDesignTime xml.sequenceOffset=-10
systemVersion	RevisionLabelString	1	attr	Version number of the System Description.

Table F.58: System

Class	SystemSignal			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreCommunication			
Note	The system signal represents the communication system's view of data exchanged between SW components which reside on different ECUs. The system signals allow to represent this communication in a flattened structure, with exactly one system signal defined for each data element prototype sent and received by connected SW component instances. Tags: atp.recommendedPackage=SystemSignals			
Base	ARElement , ARObject , CollectableElement , Identifiable , MultilanguageReferrable , PackageableElement , Referrable			
Attribute	Datatype	Mul.	Kind	Note
dynamicLength	Boolean	1	attr	The length of dynamic length signals is variable in run-time. Only a maximum length of such a signal is specified in the configuration (attribute length in ISignal element).
physicalProps	SwDataDefProps	0..1	aggr	Specification of the physical representation.

Table F.59: SystemSignal

Class	VariableInAtomicSWCTypeInstanceRef			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::DataElements::InstanceRefsUsage			
Note				
Base	ARObject , AtpInstanceRef			
Attribute	Datatype	Mul.	Kind	Note
base	AtomicSwComponentType	1	ref	Stereotypes: atpDerived Tags: xml.sequenceOffset=10
contextDataPrototype (ordered)	ApplicationCompositeElementDataPrototype	*	ref	This is the context in a compositeDataType. Tags: xml.sequenceOffset=40

Attribute	Datatype	Mul.	Kind	Note
portPrototype	PortPrototype	0..1	ref	This is the port providing the parameter or the entry point to the parameter structure. Tags: xml.sequenceOffset=20
rootVariableDataPrototype	VariableDataPrototype	0..1	ref	Tags: xml.sequenceOffset=30
targetDataPrototype	DataPrototype	1	ref	This is the target of the instance ref. Note that it shall be one of ApplicationCompositeElementDataPrototype of VariableDataPrototype. Tags: xml.sequenceOffset=50

Table F.60: VariableInAtomicSWCTypeInstanceRef

Class	VariationPoint			
Package	M2::AUTOSARTemplates::GenericStructure::VariantHandling			
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 postBuildVariantCriterion is fulfilled.			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
desc	MultiLanguageOverviewParagraph	0..1	aggr	This allows to describe shortly the purpose of the variation point. Tags: xml.sequenceOffset=20
blueprintCondition	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
formalBlueprintCondition	BlueprintFormula	0..1	aggr	This denotes a formal blueprintCondition. This shall be not in contradiction with blueprintCondition. It is recommended only to use one of the two. Tags: xml.sequenceOffset=29
postBuildVariantCondition	PostBuildVariantCondition	*	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

Attribute	Datatype	Mul.	Kind	Note
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	ref	This provides a name to the particular variation point to support the RTE generator. It is necessary for supporting splittable 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. Tags: xml.sequenceOffset=10
swSyscond	ConditionByFormula	0..1	aggr	This condition acts as Binding Function for the VariationPoint. Note that the multiplicity is 0..1 in order to support pure postBuild variants. Tags: xml.sequenceOffset=30

Table F.61: VariationPoint

Class	ImplementationDataType			
Package	M2::AUTOSARTemplates::CommonStructure::ImplementationDataTypes			
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 , AtpBlueprint , AtpBlueprintable , AtpClassifier , AtpType , AutosarDataType , CollectableElement , Identifiable , MultilanguageReferrable , PackageableElement , Referrable			
Attribute	Datatype	Mul.	Kind	Note
dynamicArraySizeProfile	String	0..1	attr	Specifies the profile which the array will follow in case this data type is a variable size array.
subElement (ordered)	ImplementationDataTypeElement	*	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

Attribute	Datatype	Mul.	Kind	Note
symbolProps	SymbolProps	0..1	aggr	This represents the SymbolProps for the ImplementationDataType. Stereotypes: atpSplitable Tags: atp.Splitkey=shortName
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 F.62: ImplementationDataType

Class	RptHook			
Package	M2::AUTOSARTemplates::SWComponentTemplate::RPTScenario			
Note	This meta class provide the ability to describe a rapid prototyping hook. This can either be described by an other AUTOSAR system with the category RPT_SYSTEM or as a non AUTOSAR software.			
Base	ARObject			
Attribute	Datatype	Mul.	Kind	Note
codeLabel	CIdentifier	0..1	ref	This attribute provides a code label which is used in the implementation of the hook. For example this can be an C function name or the name of data definition.
mcdIdentifier	NameToken	0..1	attr	This attribute provides an identifier which shall be used in a MCD System to display the Rpt Hook.
rptArHook	AtpFeature	0..1	iref	This describes the hook with the means of another AUTOSAR system.
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.

Table F.63: RptHook

G Upstream Mapping

G.1 Introduction

This chapter describes the mapping of the ECU Configuration parameters (M1 model) onto the meta-classes and attributes of the AUTOSAR upstream templates (System Template, SW Component Template and ECU Resource Template).

The relationships between upstream templates and ECU Configuration are described in order to answer typical questions like:

- How shall a supplier use the information in a System Description in order to fulfill the needs defined by the systems engineer?
- How is a tool vendor supposed to generate an ECU Configuration Description out of ECU Extract of System Description?

Please note that the tables contain the following columns:

bsw module: Name of BSW module

bsw context: Reference to parameter container

bsw type: Type of parameter

bsw param: Name of the BSW parameter

bsw desc: Description from the configuration document

m2 template: System Template, SW Component Template, ECU Resource Template

m2 param: Name of the upstream template parameter

m2 description: Description from the upstream template definition

mapping rule: Textual description on how to transform between M2 and BSW domains

mapping type:

- local: no mapping needed since parameter local to BSW
- partial: some data can be automatically mapped but not all
- full: all data can be automatically mapped

G.2 NvM

BSW Module	BSW Context	
NvM	NvM/NvMBlockDescriptor	
BSW Parameter		BSW Type
NvMBlockJobPriority		EcucIntegerParamDef

BSW Description	
Defines the job priority for a NVRAM block (0 = Immediate priority).	
Template Description	
Requires the priority of writing this block in case of concurrent requests to write other blocks.	
M2 Parameter	
CommonStructure::ServiceNeeds::NvBlockNeeds.writingPriority	
Mapping Rule	Mapping Type
It is the integrators job to secure the value-monotonic assignment of writing Priority to NvMBlockJobPriority. This means that the lowest assigned value of writingPriority=MEDIUM shall be greater than highest assigned value of writing Priority=HIGH etc.	full
Mapping Status	Mapping ID
valid	up_NvM_00016

BSW Module	BSW Context	
NvM	NvM/NvMBlockDescriptor	
BSW Parameter		BSW Type
NvMBlockManagementType		EcucEnumerationParamDef
BSW Description		
Defines the block management type for the NVRAM block.[NVM137]		
Template Description		
Reliability against data loss on the non-volatile medium.		
M2 Parameter		
CommonStructure::ServiceNeeds::NvBlockNeeds.reliability		
Mapping Rule		Mapping Type
if (reliability == errorDetection noProtection) && nDataSets==0 then NvmBlock ManagementType = NVM_BLOCK_NATIVE. if reliability == errorCorrection then NvmBlockManagementType = NVM_BLOCK_REDUNDANT. [constr_1095] applies.		full
Mapping Status		Mapping ID
valid		up_NvM_00009

BSW Module	BSW Context	
NvM	NvM/NvMBlockDescriptor	
BSW Parameter		BSW Type
NvMBlockUseAutoValidation		EcucBooleanParamDef
BSW Description		
Defines whether the RAM Block shall be auto validated during shutdown phase.		
true: if auto validation mechanism is used, false: otherwise		
Template Description		
If set to true the RAM Block shall be auto validated during shutdown phase.		
M2 Parameter		
CommonStructure::ServiceNeeds::NvBlockNeeds.useAutoValidationAtShutDown		
Mapping Rule		Mapping Type
1:1 mapping		full
Mapping Status		Mapping ID
valid		

BSW Module	BSW Context	
NvM	NvM/NvMBlockDescriptor	
BSW Parameter		BSW Type
NvMBlockUseCRCCompMechanism		EcucBooleanParamDef
BSW Description		
Defines whether the CRC of the RAM Block shall be compared during a write job with the CRC which was calculated during the last successful read or write job. true: if compare mechanism is used, false: otherwise		
Template Description		
If set to true the CRC of the RAM Block shall be compared during a write job with the CRC which was calculated during the last successful read or write job in order to skip unnecessary NVRAM writings.		
M2 Parameter		
CommonStructure::ServiceNeeds::NvBlockNeeds.useCRCCompMechanism		
Mapping Rule		Mapping Type
1:1 mapping		full
Mapping Status		Mapping ID
valid		

BSW Module	BSW Context	
NvM	NvM/NvMBlockDescriptor	
BSW Parameter		BSW Type
NvMBlockUseCrc		EcucBooleanParamDef
BSW Description		
Defines CRC usage for the NVRAM block, i.e. memory space for CRC is reserved in RAM and NV memory. true: CRC will be used for this NVRAM block. false: CRC will not be used for this NVRAM block.		
Template Description		
Reliability against data loss on the non-volatile medium.		
M2 Parameter		
CommonStructure::ServiceNeeds::NvBlockNeeds.reliability		
Mapping Rule		Mapping Type
reliability == errorCorrection errorDetection means that NvmBlockUseCrc shall be set to true, else NvmBlockUseCrc = false		full
Mapping Status		Mapping ID
valid		up_NvM_00003

BSW Module	BSW Context	
NvM	NvM/NvMBlockDescriptor	
BSW Parameter		BSW Type
NvMBlockUseSetRamBlockStatus		EcucBooleanParamDef
BSW Description		

Defines if NvMSetRamBlockStatusApi shall be used for this block or not.	
Note: If NvMSetRamBlockStatusApi is disabled this configuration parameter shall be ignored.	
true: calling of NvMSetRamBlockStatus for this RAM block shall set the status of the RAM block.	
false: calling of NvMSetRamBlockStatus for this RAM block shall be ignored.	
Template Description	
This attribute defines how the management of the RAM Block status is controlled.	
M2 Parameter	
CommonStructure::ServiceNeeds::NvBlockNeeds.ramBlockStatusControl	
Mapping Rule	Mapping Type
If the value of NvBlockNeeds.ramBlockStatusControl is set to RamBlockStatusControlEnum.api the parameter shall be set to true. If the value of NvBlockNeeds.ramBlockStatusControl is set to RamBlockStatusControlEnum.nvRamManager it shall be set to false.	full
Mapping Status	Mapping ID
valid	up_NvM_00017

BSW Module	BSW Context	
NvM	NvM/NvMBlockDescriptor	
BSW Parameter		BSW Type
NvMBlockWriteProt		EcucBooleanParamDef
BSW Description		
Defines an initial write protection of the NV block		
true: Initial block write protection is enabled. false: Initial block write protection is disabled.		
Template Description		
True: data of this NVRAM Block are write protected for normal operation (but protection can be disabled) false: no restriction		
M2 Parameter		
CommonStructure::ServiceNeeds::NvBlockNeeds.readonly		
Mapping Rule		Mapping Type
1:1 mapping		full
Mapping Status		Mapping ID
valid		up_NvM_00005

BSW Module	BSW Context	
NvM	NvM/NvMBlockDescriptor	
BSW Parameter		BSW Type
NvMCalcRamBlockCrc		EcucBooleanParamDef
BSW Description		
Defines CRC (re)calculation for the permanent RAM block or NVRAM blocks which are configured to use explicit synchronization mechanism.		
true: CRC will be (re)calculated for this permanent RAM block. false: CRC will not be (re)calculated for this permanent RAM block.		
Template Description		
Defines if CRC (re)calculation for the permanent RAM Block is required.		

M2 Parameter	
CommonStructure::ServiceNeeds::NvBlockNeeds.calcRamBlockCrc	
Mapping Rule	Mapping Type
1:1 mapping	full
Mapping Status	Mapping ID
valid	up_NvM_00007

BSW Module	BSW Context	
NvM	NvM/NvMBlockDescriptor	
BSW Parameter		BSW Type
NvMNvBlockNum		EcucIntegerParamDef
BSW Description		
Defines the number of multiple NV blocks in a contiguous area according to the given block management type. 1-255 For NVRAM blocks to be configured of block management type NVM_BLOCK_DATASET. The actual range is limited according to SWS_NvM_00444. 1 For NVRAM blocks to be configured of block management type NVM_BLOCK_NATIVE 2 For NVRAM blocks to be configured of block management type NVM_BLOCK_REDUNDANT		
Template Description		
NvBlockNeeds.nDataSets: Number of data sets to be provided by the NVRAM manager for this block. This is the total number of ROM Blocks and RAM Blocks. NvBlockNeeds.reliability: Reliability against data loss on the non-volatile medium.		
M2 Parameter		
CommonStructure::ServiceNeeds::NvBlockNeeds.nDataSets,CommonStructure::ServiceNeeds::NvBlockNeeds.reliability		
Mapping Rule		Mapping Type
if (nDataSets == 0 && reliability ==noProtection errorDetection) then NvMNvBlockNum = 1. if (nDataSets >0 && reliability ==noProtection errorDetection) then NvMNvBlockNum = nDataSets.		full
Mapping Status		Mapping ID
valid		up_NvM_00011

BSW Module	BSW Context	
NvM	NvM/NvMBlockDescriptor	
BSW Parameter		BSW Type
NvMResistantToChangedSw		EcucBooleanParamDef
BSW Description		
Defines whether a NVRAM block shall be treated resistant to configuration changes or not. If there is no default data available at configuration time then the application shall be responsible for providing the default initialization data. In this case the application has to use NvM_GetErrorStatus()to be able to distinguish between first initialization and corrupted data. true: NVRAM block is resistant to changed software. false: NVRAM block is not resistant to changed software.		
Template Description		

Defines whether an NVRAM Block shall be treated resistant to configuration changes (true) or not (false). For details how to handle initialization in the latter case, please refer to the NVRAM specification.	
M2 Parameter	
CommonStructure::ServiceNeeds::NvBlockNeeds.resistantToChangedSw	
Mapping Rule	Mapping Type
1:1 Mapping	full
Mapping Status	Mapping ID
valid	up_NvM_00006

BSW Module	BSW Context
NvM	NvM/NvMBlockDescriptor
BSW Parameter	BSW Type
NvMRomBlockNum	EcucIntegerParamDef
BSW Description	
Defines the number of multiple ROM blocks in a contiguous area according to the given block management type.	
0-254 For NVRAM blocks to be configured of block management type NVM_BLOCK_DATASET. The actual range is limited according to SWS_NvM_00444.	
0-1 For NVRAM blocks to be configured of block management type NVM_BLOCK_NATIVE	
0-1 For NVRAM blocks to be configured of block management type NVM_BLOCK_REDUNDANT	
Template Description	
Number of ROM Blocks to be provided by the NVRAM manager for this block. Please note that these multiple ROM Blocks are given in a contiguous area.	
M2 Parameter	
CommonStructure::ServiceNeeds::NvBlockNeeds.nRomBlocks	
Mapping Rule	Mapping Type
1:1 mapping	full
Mapping Status	Mapping ID
valid	up_NvM_00008

BSW Module	BSW Context
NvM	NvM/NvMBlockDescriptor
BSW Parameter	BSW Type
NvMSelectBlockForReadAll	EcucBooleanParamDef
BSW Description	
Defines whether a NVRAM block shall be processed during NvM_ReadAll or not. This configuration parameter has only influence on those NVRAM blocks which are configured to have a permanent RAM block or which are configured to use explicit synchronization mechanism.	
true: NVRAM block shall be processed by NvM_ReadAll false: NVRAM block shall not be processed by NvM_ReadAll	
Template Description	
Defines whether the associated RAM Block shall be implicitly restored during startup by the basic software.	
M2 Parameter	
CommonStructure::ServiceNeeds::NvBlockNeeds.restoreAtStart	
Mapping Rule	Mapping Type
1:1 Mapping	full

Mapping Status	Mapping ID
valid	up_NvM_00013

BSW Module	BSW Context	
NvM	NvM/NvMBlockDescriptor	
BSW Parameter		BSW Type
NvMSelectBlockForWriteAll		EcucBooleanParamDef
BSW Description		
Defines whether a NVRAM block shall be processed during NvM_WriteAll or not. This configuration parameter has only influence on those NVRAM blocks which are configured to have a permanent RAM block or which are configured to use explicit synchronization mechanism.		
true: NVRAM block shall be processed by NvM_WriteAll		
false: NVRAM block shall not be processed by NvM_WriteAll		
Template Description		
Defines whether or not the associated RAM Block shall be implicitly stored during shutdown by the basic software.		
M2 Parameter		
CommonStructure::ServiceNeeds::NvBlockNeeds.storeAtShutdown		
Mapping Rule		Mapping Type
1:1 Mapping		full
Mapping Status		Mapping ID
valid		up_NvM_00014

BSW Module	BSW Context	
NvM	NvM/NvMBlockDescriptor	
BSW Parameter		BSW Type
NvMStaticBlockIDCheck		EcucBooleanParamDef
BSW Description		
Defines if the Static Block ID check is enabled.		
false: Static Block ID check is disabled.		
true: Static Block ID check is enabled.		
Template Description		
Defines if the Static Block Id check shall be enabled.		
M2 Parameter		
CommonStructure::ServiceNeeds::NvBlockNeeds.checkStaticBlockId		
Mapping Rule		Mapping Type
1:1 mapping		full
Mapping Status		Mapping ID
valid		up_NvM_00012

BSW Module	BSW Context	
NvM	NvM/NvMBlockDescriptor	
BSW Parameter		BSW Type
NvMWriteBlockOnce		EcucBooleanParamDef
BSW Description		

Defines write protection after first write. The NVRAM manager sets the write protection bit after the NV block was written the first time. This means that some of the NV blocks in the NVRAM should never be erased nor be replaced with the default ROM data after first initialization. [NVM276]. true: Defines write protection after first write is enabled. false: Defines write protection after first write is disabled.	
Template Description	
Defines write protection after first write: true: This block is prevented from being changed/erased or being replaced with the default ROM data after first initialization by the software-component. false: No such restriction.	
M2 Parameter	
CommonStructure::ServiceNeeds::NvBlockNeeds.writeOnlyOnce	
Mapping Rule	Mapping Type
1:1 mapping	full
Mapping Status	Mapping ID
valid	up_NvM_00015

BSW Module	BSW Context	
NvM	NvM/NvMBlockDescriptor	
BSW Parameter		BSW Type
NvMWriteVerification		EcucBooleanParamDef
BSW Description		
Defines if Write Verification is enabled. false: Write verification is disabled. true: Write Verification is enabled.		
Template Description		
Defines if Write Verification shall be enabled for this NVRAM Block.		
M2 Parameter		
CommonStructure::ServiceNeeds::NvBlockNeeds.writeVerification		
Mapping Rule		Mapping Type
1:1 mapping		full
Mapping Status		Mapping ID
valid		up_NvM_00010

G.3 Com

BSW Module	BSW Context	
Com	Com/ComConfig/ComGwMapping/ComGwDestination/ComGwDestination Description	
BSW Parameter		BSW Type
ComFilter		EcucParamConfContainerDef
BSW Description		
This container contains the configuration parameters of the AUTOSAR COM module's Filters. Note: On sender side the container is used to specify the transmission mode conditions.		
Template Description		
Base class for data filters. The type of the filter is specified in attribute dataFilterType. Some of the filter types require additional arguments which are specified as attributes of this class.		
M2 Parameter		

CommonStructure::Filter::DataFilter	
Mapping Rule	Mapping Type
Create container on the receiver side if the NonqueuedReceiverComSpec contains a DataFilter. Create Container on the sender side if the TransmissionMode Condition element contains a reference to this signal.	full
Mapping Status	Mapping ID
valid	up_Com_00073

BSW Module	BSW Context	
Com	Com/ComConfig/ComGwMapping/ComGwDestination/ComGwDestinationDescription/ComFilter	
BSW Parameter		BSW Type
ComFilterAlgorithm		EcucEnumerationParamDef
BSW Description		
The range of values is specified in the [17] specification, chapter 2.2.2, Reception Filtering.		
Template Description		
This attribute specifies the type of the filter.		
M2 Parameter		
CommonStructure::Filter::DataFilter.dataFilterType		
Mapping Rule		Mapping Type
Mapping between DataFilterTypeEnum and ComFilterAlgorithm Enum is necessary.		full
Mapping Status		Mapping ID
valid		up_Com_00075

BSW Module	BSW Context	
Com	Com/ComConfig/ComGwMapping/ComGwDestination/ComGwDestinationDescription/ComFilter	
BSW Parameter		BSW Type
ComFilterMask		EcucIntegerParamDef
BSW Description		
The name of this attribute corresponds to the parameter name in the [17] specification of Reception Filtering.		
Template Description		
Mask for old and new value.		
M2 Parameter		
CommonStructure::Filter::DataFilter.mask		
Mapping Rule		Mapping Type
1:1 mapping		full
Mapping Status		Mapping ID
valid		up_Com_00078

BSW Module	BSW Context	
Com	Com/ComConfig/ComGwMapping/ComGwDestination/ComGwDestinationDescription/ComFilter	
BSW Parameter		BSW Type
ComFilterMax		EcucIntegerParamDef
BSW Description		
The name of this attribute corresponds to the parameter name in the [17] specification of Reception Filtering.		
Template Description		

Value to specify the upper boundary	
M2 Parameter	
CommonStructure::Filter::DataFilter.max	
Mapping Rule	Mapping Type
1:1 mapping	full
Mapping Status	Mapping ID
valid	up_Com_00077

BSW Module	BSW Context	
Com	Com/ComConfig/ComGwMapping/ComGwDestination/ComGwDestinationDescription/ComFilter	
BSW Parameter		BSW Type
ComFilterMin		EcucIntegerParamDef
BSW Description		
The name of this attribute corresponds to the parameter name in the [17] specification of Reception Filtering.		
Template Description		
Value to specify the lower boundary		
M2 Parameter		
CommonStructure::Filter::DataFilter.min		
Mapping Rule	Mapping Type	
1:1 mapping	full	
Mapping Status	Mapping ID	
valid	up_Com_00080	

BSW Module	BSW Context	
Com	Com/ComConfig/ComGwMapping/ComGwDestination/ComGwDestinationDescription/ComFilter	
BSW Parameter		BSW Type
ComFilterOffset		EcucIntegerParamDef
BSW Description		
The name of this attribute corresponds to the parameter name in the [17] specification of Reception Filtering.		
Range = 0..(ComFilterPeriod-1)		
Template Description		
Specifies the initial number of messages to occur before the first message is passed		
M2 Parameter		
CommonStructure::Filter::DataFilter.offset		
Mapping Rule	Mapping Type	
1:1 mapping	full	
Mapping Status	Mapping ID	
valid	up_Com_00076	

BSW Module	BSW Context	
Com	Com/ComConfig/ComGwMapping/ComGwDestination/ComGwDestinationDescription/ComFilter	
BSW Parameter		BSW Type
ComFilterPeriod		EcucIntegerParamDef
BSW Description		

This parameter defines the period of the ComFilterAlgorithm ONE_EVERY_N.	
Template Description	
Specifies number of messages to occur before the message is passed again	
M2 Parameter	
CommonStructure::Filter::DataFilter.period	
Mapping Rule	Mapping Type
1:1 mapping	full
Mapping Status	Mapping ID
valid	up_Com_00079

BSW Module	BSW Context	
Com	Com/ComConfig/ComGwMapping/ComGwDestination/ComGwDestinationDescription/ComFilter	
BSW Parameter		BSW Type
ComFilterX		EcucIntegerParamDef
BSW Description		
The name of this attribute corresponds to the parameter name in the [17] specification of Reception Filtering.		
Template Description		
Value to compare with		
M2 Parameter		
CommonStructure::Filter::DataFilter.x		
Mapping Rule		Mapping Type
1:1 mapping		full
Mapping Status		Mapping ID
valid		up_Com_00074

BSW Module	BSW Context	
Com	Com/ComConfig/ComGwMapping/ComGwDestination/ComGwDestinationDescription	
BSW Parameter		BSW Type
ComSignalInitValue		EcucStringParamDef
BSW Description		
Initial value for this signal. In case of UINT8_N the default value is a string of length ComSignalLength with all bytes set to 0x00. In case of UINT8_DYN the initial size shall be 0.		
In case the ComSignalType is UINT8, UINT16, UINT32, UINT64, SINT8, SINT16, SINT32, SINT64 the string shall be interpreted as defined in the chapter Integer Type in the AUTOSAR EcuC specification.		
In case the ComSignalType is FLOAT32, FLOAT64 the string shall be interpreted as defined in the chapter Float Type in the AUTOSAR EcuC specification.		
In case the ComSignalType is BOOLEAN the string shall be interpreted as defined in the chapter Boolean Type in the AUTOSAR EcuC specification.		
In case the ComSignal is a UINT8_N, UINT8_DYN the string shall be interpreted as a decimal representation of the characters separated by blanks, e.g. "97 98 100" means a string "abd", where the char "a" is in byte 0(lowest address), "b" is in byte 1, and "d" is in byte 2 and (highest address). For the ComSignalType UINT8_DYN the dynamic length shall be set to the number of configured characters. An empty string "" shall be interpreted as 0-sized dynamic signal.		
Template Description		
Initial value to be sent if sender component is not yet fully initialized, but receiver needs data already.		
M2 Parameter		

SystemTemplate::Fibex::FibexCore::CoreCommunication::ISignal.initValue,SWComponentTemplate::Communication::NonqueuedSenderComSpec.initValue	
Mapping Rule	Mapping Type
It is possible to aggregate an initValue at the level of a ComSpec in the SW C Template. in case the System Description doesn't use a complete Software Component Description (VFB View) the initValue is defined in the System Template.	full
Mapping Status	Mapping ID
valid	up_Com_00081

BSW Module	BSW Context	
Com	Com/ComConfig/ComSignal	
BSW Parameter		BSW Type
ComDataInvalidAction		EcucEnumerationParamDef
BSW Description		
This parameter defines the action performed upon reception of an invalid signal. Relating to signal groups the action in case if one of the included signals is an invalid signal. If Replace is used the ComSignalInitValue will be used for the replacement.		
Template Description		
Specifies whether the component can actively invalidate a particular dataElement.		
If no invalidationPolicy points to a dataElement this is considered to yield the identical result as if the handleInvalid attribute was set to dontInvalidate.		
M2 Parameter		
SWComponentTemplate::PortInterface::InvalidationPolicy		
Mapping Rule		Mapping Type
If strategy HandleInvalidEnum.keep is defined then set parameter to notify. If strategy HandleInvalidEnum.replace is defined then set parameter to replace. If the parameter does not exist this corresponds to the value HandleInvalidEnum.dontInvalidate.		full
Mapping Status		Mapping ID
valid		up Com 00071

BSW Module		BSW Context	
Com		Com/ComConfig/ComSignal	
BSW Parameter		BSW Type	
ComFilter		EcucParamConfContainerDef	
BSW Description			
This container contains the configuration parameters of the AUTOSAR COM module's Filters.			
Note: On sender side the container is used to specify the transmission mode conditions.			
Template Description			
Base class for data filters. The type of the filter is specified in attribute dataFilterType. Some of the filter types require additional arguments which are specified as attributes of this class.			
M2 Parameter			
CommonStructure::Filter::DataFilter			
Mapping Rule			Mapping Type
Create container on the receiver side if the NonqueuedReceiverComSpec contains a DataFilter. Create Container on the sender side if the TransmissionMode Condition element contains a reference to this signal.			full
Mapping Status			Mapping ID
valid			up_Com_00073

BSW Module	BSW Context	
Com	Com/ComConfig/ComSignal/ComFilter	
BSW Parameter		BSW Type
ComFilterAlgorithm		EcucEnumerationParamDef
BSW Description		
The range of values is specified in the [17] specification, chapter 2.2.2, Reception Filtering.		
Template Description		
This attribute specifies the type of the filter.		
M2 Parameter		
CommonStructure::Filter::DataFilter.dataFilterType		
Mapping Rule		Mapping Type
Mapping between DataFilterTypeEnum and ComFilterAlgorithm Enum is necessary.		full
Mapping Status		Mapping ID
valid		up_Com_00075

BSW Module	BSW Context	
Com	Com/ComConfig/ComSignal/ComFilter	
BSW Parameter		BSW Type
ComFilterMask		EcucIntegerParamDef
BSW Description		
The name of this attribute corresponds to the parameter name in the [17] specification of Reception Filtering.		
Template Description		
Mask for old and new value.		
M2 Parameter		
CommonStructure::Filter::DataFilter.mask		
Mapping Rule		Mapping Type
1:1 mapping		full
Mapping Status		Mapping ID
valid		up_Com_00078

BSW Module	BSW Context	
Com	Com/ComConfig/ComSignal/ComFilter	
BSW Parameter		BSW Type
ComFilterMax		EcucIntegerParamDef
BSW Description		
The name of this attribute corresponds to the parameter name in the [17] specification of Reception Filtering.		
Template Description		
Value to specify the upper boundary		
M2 Parameter		
CommonStructure::Filter::DataFilter.max		
Mapping Rule		Mapping Type
1:1 mapping		full
Mapping Status		Mapping ID
valid		up_Com_00077

BSW Module	BSW Context	
Com	Com/ComConfig/ComSignal/ComFilter	

BSW Parameter		BSW Type
ComFilterMin		EcucIntegerParamDef
BSW Description		
The name of this attribute corresponds to the parameter name in the [17] specification of Reception Filtering.		
Template Description		
Value to specify the lower boundary		
M2 Parameter		
CommonStructure::Filter::DataFilter.min		
Mapping Rule		Mapping Type
1:1 mapping		full
Mapping Status		Mapping ID
valid		up_Com_00080

BSW Module	BSW Context	
Com	Com/ComConfig/ComSignal/ComFilter	
BSW Parameter		BSW Type
ComFilterOffset		EcucIntegerParamDef
BSW Description		
The name of this attribute corresponds to the parameter name in the [17] specification of Reception Filtering.		
Range = 0..(ComFilterPeriod-1)		
Template Description		
Specifies the initial number of messages to occur before the first message is passed		
M2 Parameter		
CommonStructure::Filter::DataFilter.offset		
Mapping Rule		Mapping Type
1:1 mapping		full
Mapping Status		Mapping ID
valid		up_Com_00076

BSW Module	BSW Context	
Com	Com/ComConfig/ComSignal/ComFilter	
BSW Parameter		BSW Type
ComFilterPeriod		EcucIntegerParamDef
BSW Description		
This parameter defines the period of the ComFilterAlgorithm ONE_EVERY_N.		
Template Description		
Specifies number of messages to occur before the message is passed again		
M2 Parameter		
CommonStructure::Filter::DataFilter.period		
Mapping Rule		Mapping Type
1:1 mapping		full
Mapping Status		Mapping ID
valid		up_Com_00079

BSW Module	BSW Context	
Com	Com/ComConfig/ComSignal/ComFilter	
BSW Parameter		BSW Type

ComFilterX	EcucIntegerParamDef
BSW Description	
The name of this attribute corresponds to the parameter name in the [17] specification of Reception Filtering.	
Template Description	
Value to compare with	
M2 Parameter	
CommonStructure::Filter::DataFilter.x	
Mapping Rule	Mapping Type
1:1 mapping	full
Mapping Status	Mapping ID
valid	up_Com_00074

BSW Module	BSW Context	
Com	Com/ComConfig/ComSignal	
BSW Parameter		BSW Type
ComSignalDataInvalidValue		EcucStringParamDef
BSW Description		
Defines the data invalid value of the signal.		
In case the ComSignalType is UINT8, UINT16, UINT32, UINT64, SINT8, SINT16, SINT32, SINT64 the string shall be interpreted as defined in the chapter Integer Type in the AUTOSAR EcuC specification.		
In case the ComSignalType is FLOAT32, FLOAT64 the string shall be interpreted as defined in the chapter Float Type in the AUTOSAR EcuC specification.		
In case the ComSignalType is BOOLEAN the string shall be interpreted as defined in the chapter Boolean Type in the AUTOSAR EcuC specification.		
In case the ComSignal is a UINT8_N, UINT8_DYN the string shall be interpreted as a decimal representation of the characters separated by blanks, e.g. "97 98 100" means a string "abd", where the char "a" is in byte 0(lowest address), "b" is in byte 1, and "d" is in byte 2 and (highest address). For the ComSignalType UINT8_DYN the dynamic length shall be set to the number of configured characters. An empty string "" shall be interpreted as 0-sized dynamic signal.		
Template Description		
Optional value to express invalidity of the actual data element.		
M2 Parameter		
DataDictionary::DataDefProperties::SwDataDefProps.invalidValue		
Mapping Rule		Mapping Type
1:1 mapping		full
Mapping Status		Mapping ID
valid		up_Com_00067

BSW Module	BSW Context	
Com	Com/ComConfig/ComSignal	
BSW Parameter		BSW Type
ComSignalInitValue		EcucStringParamDef
BSW Description		

Initial value for this signal. In case of UINT8_N the default value is a string of length ComSignalLength with all bytes set to 0x00. In case of UINT8_DYN the initial size shall be 0.	
In case the ComSignalType is UINT8, UINT16, UINT32, UINT64, SINT8, SINT16, SINT32, SINT64 the string shall be interpreted as defined in the chapter Integer Type in the AUTOSAR EcuC specification.	
In case the ComSignalType is FLOAT32, FLOAT64 the string shall be interpreted as defined in the chapter Float Type in the AUTOSAR EcuC specification.	
In case the ComSignalType is BOOLEAN the string shall be interpreted as defined in the chapter Boolean Type in the AUTOSAR EcuC specification.	
In case the ComSignal is a UINT8_N, UINT8_DYN the string shall be interpreted as a decimal representation of the characters separated by blanks, e.g. "97 98 100" means a string "abd", where the char "a" is in byte 0(lowest address), "b" is in byte 1, and "d" is in byte 2 and (highest address). For the ComSignalType UINT8_DYN the dynamic length shall be set to the number of configured characters. An empty string "" shall be interpreted as 0-sized dynamic signal.	
Template Description	
Initial value to be sent if sender component is not yet fully initialized, but receiver needs data already.	
M2 Parameter	
SystemTemplate::Fibex::FibexCore::CoreCommunication::ISignal.initValue,SWComponentTemplate::Communication::NonqueuedSenderComSpec.initValue	
Mapping Rule	Mapping Type
It is possible to aggregate an initValue at the level of a ComSpec in the SW C Template. in case the System Description doesn't use a complete Software Component Description (VFB View) the initValue is defined in the System Template.	full
Mapping Status	Mapping ID
valid	up_Com_00081

BSW Module	BSW Context
Com	Com/ComConfig/ComSignal
BSW Parameter	BSW Type
ComTimeout	EcucFloatParamDef
BSW Description	
Defines the length of the deadline monitoring timeout period in seconds. The period for the first timeout period can be configured separately by ECUC_Com_00183.	
Template Description	
NonqueuedReceiverComSpec.aliveTimeout:	
Specify the amount of time (in seconds) after which the software component (via the RTE) needs to be notified if the corresponding data item have not been received according to the specified timing description.	
If the aliveTimeout attribute is 0 no timeout monitoring shall be performed.	
ISignalPort.timeout:	
Optional timeout value in seconds for the reception of the ISignal.	
In case the System Description doesn't use a complete Software Component Description (VFB View). This supports the inclusion of legacy system signals.	
If a full DataMapping exist for the SystemSignal this information may be available from a configured ReceiverComSpec, in this case the timeout value in ReceiverComSpec override this optional timeout specification.	
M2 Parameter	
SWComponentTemplate::Communication::NonqueuedReceiverComSpec.aliveTimeout,SystemTemplate::Fibex::FibexCore::CoreCommunication::ISignalPort.timeout	

Mapping Rule	Mapping Type
If a full DataMapping exist for the SystemSignal this information may be available from a configured NonqueuedReceiverComSpec. In this case the timeout value in ReceiverComSpec overrides the optional timeout specification in the System Template. Please note that the SWS_RTE defines an algorithm to finally set the applicable timeout value.	full
Mapping Status	Mapping ID
valid	up_Com_00068

BSW Module	BSW Context	
Com	Com/ComConfig/ComSignalGroup	
BSW Parameter		BSW Type
ComDataInvalidAction		EcucEnumerationParamDef
BSW Description		
This parameter defines the action performed upon reception of an invalid signal. Relating to signal groups the action in case if one of the included signals is an invalid signal. If Replace is used the ComSignalInitValue will be used for the replacement.		
Template Description		
Specifies whether the component can actively invalidate a particular dataElement.		
If no invalidationPolicy points to a dataElement this is considered to yield the identical result as if the handleInvalid attribute was set to dontInvalidate.		
M2 Parameter		
SWComponentTemplate::PortInterface::InvalidationPolicy		
Mapping Rule	Mapping Type	
If strategy HandleInvalidEnum.keep is defined then set parameter to notify. If strategy HandleInvalidEnum.replace is defined then set parameter to replace. If the parameter does not exist this corresponds to the value HandleInvalidEnum.dontInvalidate.	full	
Mapping Status	Mapping ID	
valid	up_Com_00071	

BSW Module	BSW Context	
Com	Com/ComConfig/ComSignalGroup/ComGroupSignal	
BSW Parameter		BSW Type
ComFilter		EcucParamConfContainerDef
BSW Description		
This container contains the configuration parameters of the AUTOSAR COM module's Filters.		
Note: On sender side the container is used to specify the transmission mode conditions.		
Template Description		
Base class for data filters. The type of the filter is specified in attribute dataFilterType. Some of the filter types require additional arguments which are specified as attributes of this class.		
M2 Parameter		
CommonStructure::Filter::DataFilter		
Mapping Rule	Mapping Type	
Create container on the receiver side if the NonqueuedReceiverComSpec contains a DataFilter. Create Container on the sender side if the TransmissionMode Condition element contains a reference to this signal.	full	
Mapping Status	Mapping ID	
valid	up_Com_00073	

BSW Module	BSW Context	
Com	Com/ComConfig/ComSignalGroup/ComGroupSignal/ComFilter	
BSW Parameter		BSW Type
ComFilterAlgorithm		EcucEnumerationParamDef
BSW Description		
The range of values is specified in the [17] specification, chapter 2.2.2, Reception Filtering.		
Template Description		
This attribute specifies the type of the filter.		
M2 Parameter		
CommonStructure::Filter::DataFilter.dataFilterType		
Mapping Rule		Mapping Type
Mapping between DataFilterTypeEnum and ComFilterAlgorithm Enum is necessary.		full
Mapping Status		Mapping ID
valid		up_Com_00075

BSW Module	BSW Context	
Com	Com/ComConfig/ComSignalGroup/ComGroupSignal/ComFilter	
BSW Parameter		BSW Type
ComFilterMask		EcucIntegerParamDef
BSW Description		
The name of this attribute corresponds to the parameter name in the [17] specification of Reception Filtering.		
Template Description		
Mask for old and new value.		
M2 Parameter		
CommonStructure::Filter::DataFilter.mask		
Mapping Rule		Mapping Type
1:1 mapping		full
Mapping Status		Mapping ID
valid		up_Com_00078

BSW Module	BSW Context	
Com	Com/ComConfig/ComSignalGroup/ComGroupSignal/ComFilter	
BSW Parameter		BSW Type
ComFilterMax		EcucIntegerParamDef
BSW Description		
The name of this attribute corresponds to the parameter name in the [17] specification of Reception Filtering.		
Template Description		
Value to specify the upper boundary		
M2 Parameter		
CommonStructure::Filter::DataFilter.max		
Mapping Rule		Mapping Type
1:1 mapping		full
Mapping Status		Mapping ID
valid		up_Com_00077

BSW Module	BSW Context	
Com	Com/ComConfig/ComSignalGroup/ComGroupSignal/ComFilter	

BSW Parameter		BSW Type
ComFilterMin		EcucIntegerParamDef
BSW Description		
The name of this attribute corresponds to the parameter name in the [17] specification of Reception Filtering.		
Template Description		
Value to specify the lower boundary		
M2 Parameter		
CommonStructure::Filter::DataFilter.min		
Mapping Rule		Mapping Type
1:1 mapping		full
Mapping Status		Mapping ID
valid		up_Com_00080

BSW Module	BSW Context	
Com	Com/ComConfig/ComSignalGroup/ComGroupSignal/ComFilter	
BSW Parameter		BSW Type
ComFilterOffset		EcucIntegerParamDef
BSW Description		
The name of this attribute corresponds to the parameter name in the [17] specification of Reception Filtering.		
Range = 0..(ComFilterPeriod-1)		
Template Description		
Specifies the initial number of messages to occur before the first message is passed		
M2 Parameter		
CommonStructure::Filter::DataFilter.offset		
Mapping Rule		Mapping Type
1:1 mapping		full
Mapping Status		Mapping ID
valid		up_Com_00076

BSW Module	BSW Context	
Com	Com/ComConfig/ComSignalGroup/ComGroupSignal/ComFilter	
BSW Parameter		BSW Type
ComFilterPeriod		EcucIntegerParamDef
BSW Description		
This parameter defines the period of the ComFilterAlgorithm ONE_EVERY_N.		
Template Description		
Specifies number of messages to occur before the message is passed again		
M2 Parameter		
CommonStructure::Filter::DataFilter.period		
Mapping Rule		Mapping Type
1:1 mapping		full
Mapping Status		Mapping ID
valid		up_Com_00079

BSW Module	BSW Context	
Com	Com/ComConfig/ComSignalGroup/ComGroupSignal/ComFilter	
BSW Parameter		BSW Type

ComFilterX	EcucIntegerParamDef
BSW Description	
The name of this attribute corresponds to the parameter name in the [17] specification of Reception Filtering.	
Template Description	
Value to compare with	
M2 Parameter	
CommonStructure::Filter::DataFilter.x	
Mapping Rule	Mapping Type
1:1 mapping	full
Mapping Status	Mapping ID
valid	up_Com_00074

BSW Module	BSW Context
Com	Com/ComConfig/ComSignalGroup/ComGroupSignal
BSW Parameter	BSW Type
ComSignalDataInvalidValue	EcucStringParamDef
BSW Description	
Defines the data invalid value of the signal.	
In case the ComSignalType is UINT8, UINT16, UINT32, UINT64, SINT8, SINT16, SINT32, SINT64 the string shall be interpreted as defined in the chapter Integer Type in the AUTOSAR EcuC specification. In case the ComSignalType is FLOAT32, FLOAT64 the string shall be interpreted as defined in the chapter Float Type in the AUTOSAR EcuC specification. In case the ComSignalType is BOOLEAN the string shall be interpreted as defined in the chapter Boolean Type in the AUTOSAR EcuC specification. In case the ComSignal is a UINT8_N, UINT8_DYN the string shall be interpreted as a decimal representation of the characters separated by blanks, e.g. "97 98 100" means a string "abd", where the char "a" is in byte 0(lowest address), "b" is in byte 1, and "d" is in byte 2 and (highest address). For the ComSignalType UINT8_DYN the dynamic length shall be set to the number of configured characters. An empty string "" shall be interpreted as 0-sized dynamic signal.	
Template Description	
Optional value to express invalidity of the actual data element.	
M2 Parameter	
DataDictionary::DataDefProperties::SwDataDefProps.invalidValue	
Mapping Rule	Mapping Type
1:1 mapping	full
Mapping Status	Mapping ID
valid	up_Com_00067

BSW Module	BSW Context
Com	Com/ComConfig/ComSignalGroup/ComGroupSignal
BSW Parameter	BSW Type
ComSignalInitValue	EcucStringParamDef
BSW Description	

Initial value for this signal. In case of UINT8_N the default value is a string of length ComSignalLength with all bytes set to 0x00. In case of UINT8_DYN the initial size shall be 0.	
In case the ComSignalType is UINT8, UINT16, UINT32, UINT64, SINT8, SINT16, SINT32, SINT64 the string shall be interpreted as defined in the chapter Integer Type in the AUTOSAR EcuC specification.	
In case the ComSignalType is FLOAT32, FLOAT64 the string shall be interpreted as defined in the chapter Float Type in the AUTOSAR EcuC specification.	
In case the ComSignalType is BOOLEAN the string shall be interpreted as defined in the chapter Boolean Type in the AUTOSAR EcuC specification.	
In case the ComSignal is a UINT8_N, UINT8_DYN the string shall be interpreted as a decimal representation of the characters separated by blanks, e.g. "97 98 100" means a string "abd", where the char "a" is in byte 0(lowest address), "b" is in byte 1, and "d" is in byte 2 and (highest address). For the ComSignalType UINT8_DYN the dynamic length shall be set to the number of configured characters. An empty string "" shall be interpreted as 0-sized dynamic signal.	
Template Description	
Initial value to be sent if sender component is not yet fully initialized, but receiver needs data already.	
M2 Parameter	
SystemTemplate::Fibex::FibexCore::CoreCommunication::ISignal.initValue,SWComponentTemplate::Communication::NonqueuedSenderComSpec.initValue	
Mapping Rule	Mapping Type
It is possible to aggregate an initValue at the level of a ComSpec in the SW C Template. in case the System Description doesn't use a complete Software Component Description (VFB View) the initValue is defined in the System Template.	full
Mapping Status	Mapping ID
valid	up_Com_00081

BSW Module	BSW Context
Com	Com/ComConfig/ComSignalGroup
BSW Parameter	BSW Type
ComTimeout	EcucFloatParamDef
BSW Description	
Defines the length of the deadline monitoring timeout period in seconds. The period for the first timeout period can be configured separately by ECUC_Com_00183.	
Template Description	
NonqueuedReceiverComSpec.aliveTimeout:	
Specify the amount of time (in seconds) after which the software component (via the RTE) needs to be notified if the corresponding data item have not been received according to the specified timing description.	
If the aliveTimeout attribute is 0 no timeout monitoring shall be performed.	
ISignalPort.timeout:	
Optional timeout value in seconds for the reception of the ISignal.	
In case the System Description doesn't use a complete Software Component Description (VFB View). This supports the inclusion of legacy system signals.	
If a full DataMapping exist for the SystemSignal this information may be available from a configured ReceiverComSpec, in this case the timeout value in ReceiverComSpec override this optional timeout specification.	
M2 Parameter	
SWComponentTemplate::Communication::NonqueuedReceiverComSpec.aliveTimeout,SystemTemplate::Fibex::FibexCore::CoreCommunication::ISignalPort.timeout	

Mapping Rule	Mapping Type
If a full DataMapping exist for the SystemSignal this information may be available from a configured NonqueuedReceiverComSpec. In this case the timeout value in ReceiverComSpec overrides the optional timeout specification in the System Template. Please note that the SWS_RTE defines an algorithm to finally set the applicable timeout value.	full
Mapping Status	Mapping ID
valid	up_Com_00068

G.4 WdgM

BSW Module	BSW Context	
WdgM	WdgM/WdgMConfigSet/WdgMMode/WdgMLocalStatusParams	
BSW Parameter		BSW Type
WdgMFailedAliveSupervisionRefCycleTol		EcucIntegerParamDef
BSW Description		
This parameter shall contain the acceptable amount of reference cycles with incorrect/failed alive supervisions for this Supervised Entity.		
Template Description		
Number of consecutive failed alive cycles for this SupervisedEntity which shall be tolerated until the supervision status of the SupervisedEntity is set to WDG_M_ALIVE_EXPIRED (see SWS WdgM for more details).		
Note that this value has to be recalculated with respect to the WdgM's own cycle time for ECU configuration.		
M2 Parameter		
CommonStructure::ServiceNeeds::SupervisedEntityNeeds.toleratedFailedCycles		
Mapping Rule		Mapping Type
1:1		full
Mapping Status		Mapping ID
valid		up_WdgM_00001

G.5 Dcm

BSW Module	BSW Context	
Dcm	Dcm/DcmConfigSet/DcmDsp/DcmDspData	
BSW Parameter		BSW Type
DcmDspDataUsePort		EcucEnumerationParamDef
BSW Description		
Defines which interface shall be used to access the data.		
Template Description		
This attribute controls whether interaction requires the software-component to react synchronously on a request or whether it processes the request in background but still the DCM has to issue the call again to eventually obtain the result of the request.		
M2 Parameter		
CommonStructure::ServiceNeeds::DiagnosticValueNeeds.processingStyle		
Mapping Rule		Mapping Type
1:1 mapping		full
Mapping Status		Mapping ID
valid		up_Dcm_00001

BSW Module	BSW Context	
Dcm	Dcm/DcmConfigSet/DcmDsp/DcmDspData/DcmDspDataUsePort	
BSW Parameter		BSW Type
USE_DATA_ASYNC_CLIENT_SERVER		EcucEnumerationLiteralDef
BSW Description		
The DCM will access the Data using an R-Port requiring a asynchronous ClientServerInterface DataServices_{Data}. The R-Port is named DataServices_{Data} where {Data} is the name of the container DcmDspData.		
Template Description		
The software-component processes the request in background but still the DCM has to issue the call again to eventually obtain the result of the request.		
M2 Parameter		
CommonStructure::ServiceNeeds::DiagnosticProcessingStyleEnum.processingStyleAsynchronous		
Mapping Rule		Mapping Type
DiagnosticServiceSwMapping is having a SwcServiceDependency and ServiceNeeds::DiagnosticProcessingStyleEnum is equal to processingStyleAsynchronous		full
Mapping Status		Mapping ID
valid		

BSW Module	BSW Context	
Dcm	Dcm/DcmConfigSet/DcmDsp/DcmDspData/DcmDspDataUsePort	
BSW Parameter		BSW Type
USE_DATA_ASYNC_CLIENT_SERVER_ERROR		EcucEnumerationLiteralDef
BSW Description		
The Dcm will access the Data using an R-Port requiring a asynchronous ClientServerInterface DataServices_{Data}. The parameter ErrorCode can be returned to allow the application to trigger a negative response during the operation. The R-Port is named DataServices_{Data} where {Data} is the name of the container DcmDspData.		
Template Description		
The software-component is superposed to react synchronously on the request.		
M2 Parameter		
CommonStructure::ServiceNeeds::DiagnosticProcessingStyleEnum.processingStyleSynchronous		
Mapping Rule		Mapping Type
DiagnosticServiceSwMapping is having a SwcServiceDependency and ServiceNeeds::DiagnosticProcessingStyleEnum is equal to processingStyleSynchronous		full
Mapping Status		Mapping ID
valid		

BSW Module	BSW Context	
Dcm	Dcm/DcmConfigSet/DcmDsp/DcmDspData/DcmDspDataUsePort	
BSW Parameter		BSW Type
USE_DATA_ASYNC_FNC_ERROR		EcucEnumerationLiteralDef
BSW Description		
The DCM will access the Data using the functions that are defined in parameters of type EcucFunctionNameDef (but without DcmDspDataReadDataLengthFnc) in the DcmDspData container. DCM_E_PENDING return is allowed. OpStatus is existing as IN parameter. The parameter ErrorCode can be returned to allow the application to trigger a negative response during the operation.		
Template Description		
The software-component is superposed to react synchronously on the request.		
M2 Parameter		

CommonStructure::ServiceNeeds::DiagnosticProcessingStyleEnum.processingStyleSynchronous	
Mapping Rule	Mapping Type
DiagnosticServiceSwMapping is having a SwcServiceDependency and ServiceNeeds::DiagnosticProcessingStyleEnum is equal to processingStyleSynchronous	full
Mapping Status	Mapping ID
valid	

BSW Module	BSW Context
Dcm	Dcm/DcmConfigSet/DcmDsp/DcmDspData/DcmDspDataUsePort
BSW Parameter	BSW Type
USE_DATA_SYNCH_CLIENT_SERVER	EcucEnumerationLiteralDef
BSW Description	
The DCM will access the Data using an R-Port requiring a synchronous ClientServerInterface DataServices_{Data}.	
The R-Port is named DataServices_{Data} where {Data} is the name of the container DcmDspData.	
Template Description	
The software-component is superposed to react synchronously on the request.	
M2 Parameter	
CommonStructure::ServiceNeeds::DiagnosticProcessingStyleEnum.processingStyleSynchronous	
Mapping Rule	Mapping Type
DiagnosticServiceSwMapping is having a SwcServiceDependency and ServiceNeeds::DiagnosticProcessingStyleEnum is equal to processingStyleSynchronous	full
Mapping Status	Mapping ID
valid	

BSW Module	BSW Context
Dcm	Dcm/DcmConfigSet/DcmDsp/DcmDspDidInfo/DcmDspDidControl
BSW Parameter	BSW Type
DcmDspDidFreezeCurrentState	EcucBooleanParamDef
BSW Description	
This indicates the presence of "FreezeCurrentState".	
Template Description	
DiagnosticIOControl.freezeCurrentState:	
Setting this attribute to true represents the ability of the Dcm to execute a freezeCurrentState.	
DiagnosticIoControlNeeds.freezeCurrentStateSupported:	
This attribute determines, if the referenced port supports temporary freezing of I/O value.	
M2 Parameter	
DiagnosticExtract::Dcm::DiagnosticService::IOControl::DiagnosticIOControl.freezeCurrentState, CommonStructure::ServiceNeeds::DiagnosticIoControlNeeds.freezeCurrentStateSupported	
Mapping Rule	Mapping Type
1:1 mapping	full
Mapping Status	Mapping ID
valid	

BSW Module	BSW Context
Dcm	Dcm/DcmConfigSet/DcmDsp/DcmDspDidInfo/DcmDspDidControl
BSW Parameter	BSW Type
DcmDspDidResetToDefault	EcucBooleanParamDef

BSW Description	
This indicates the presence of "ResetToDefault".	
Template Description	
DiagnosticIOControl.resetToDefault: Setting this attribute to true represents the ability of the Dcm to execute a resetToDefault.	
DiagnosticIOControlNeeds.resetToDefaultSupported: This represents a flag for the existence of the ResetToDefault operation in the service interface.	
M2 Parameter	
DiagnosticExtract::Dcm::DiagnosticService::IOControl::DiagnosticIOControl.resetToDefault, CommonStructure::ServiceNeeds::DiagnosticIOControlNeeds.resetToDefaultSupported	
Mapping Rule	Mapping Type
1:1 mapping	full
Mapping Status	Mapping ID
valid	

BSW Module	BSW Context
Dcm	Dcm/DcmConfigSet/DcmDsp/DcmDspDidInfo/DcmDspDidControl
BSW Parameter	BSW Type
DcmDspDidShortTermAdjustment	EcucBooleanParamDef
BSW Description	
This indicates the presence of "ShortTermAdjustment".	
Template Description	
DiagnosticIOControl.shortTermAdjustment: Setting this attribute to true represents the ability of the Dcm to execute a shortTermAdjustment.	
DiagnosticIOControlNeeds.shortTermAdjustmentSupported: This attribute determines, if the referenced port supports temporarily setting of I/O value to a specific value provided by the diagnostic tester.	
M2 Parameter	
DiagnosticExtract::Dcm::DiagnosticService::IOControl::DiagnosticIOControl.shortTermAdjustment, CommonStructure::ServiceNeeds::DiagnosticIOControlNeeds.shortTermAdjustmentSupported	
Mapping Rule	Mapping Type
1:1 mapping	full
Mapping Status	Mapping ID
valid	

H Splitable Elements in the Scope of this Document

This chapter contains a table of all model elements stereotyped «atpSplitable» in the scope of this document.

Each entry in Table H.1 consists of the identification of the specific model element itself and the applicable value of the tagged value `atp.Splitkey`.

For more information about the concept of splitable model elements and how these shall be treated please refer to [12].

Name of splitable element	Splitkey
ARPackage.arPackage	shortName, variationPoint.shortLabel
ARPackage.element	shortName, variationPoint.shortLabel
ARPackage.referenceBase	shortLabel
AtomicSwComponentType.internalBehavior	internalBehavior, variationPoint.shortLabel
AtomicSwComponentType.symbolProps	shortName
CompositionSwComponentType.component	shortName, variationPoint.shortLabel
CompositionSwComponentType.connector	shortName, variationPoint.shortLabel
CompositionSwComponentType.constantValueMapping	constantValueMapping
CompositionSwComponentType.dataTypeMapping	dataTypeMapping
CompositionSwComponentType.instantiationRTEEvent-Props	shortLabel, variationPoint.shortLabel
ConsistencyNeeds.dpgDoesNotRequireCoherency	shortName, variationPoint.shortLabel
ConsistencyNeeds.dpgRequiresCoherency	shortName, variationPoint.shortLabel
ConsistencyNeeds.regDoesNotRequireStability	shortName, variationPoint.shortLabel
ConsistencyNeeds.regRequiresStability	shortName, variationPoint.shortLabel
EndToEndProtection.endToEndProfile	endToEndProfile
EndToEndProtection.endToEndProtectionISignalIPdu	variationPoint.shortLabel
EndToEndProtection.endToEndProtectionVariablePro- totype	shortLabel, variationPoint.shortLabel
EndToEndProtectionSet.endToEndProtection	shortName, variationPoint.shortLabel
Implementation.mcSupport	mcSupport
Implementation.resourceConsumption	shortName
ImplementationDataType.symbolProps	shortName
InternalBehavior.constantMemory	shortName, variationPoint.shortLabel
InternalBehavior.constantValueMapping	constantValueMapping
InternalBehavior.dataTypeMapping	dataTypeMapping
InternalBehavior.exclusiveArea	shortName, variationPoint.shortLabel
InternalBehavior.exclusiveAreaNestingOrder	shortName, variationPoint.shortLabel
InternalBehavior.staticMemory	shortName, variationPoint.shortLabel
NvBlockDescriptor.constantValueMapping	constantValueMapping
NvBlockDescriptor.dataTypeMapping	dataTypeMapping
NvBlockSwComponentType.nvBlockDescriptor	shortName, variationPoint.shortLabel
ParameterSwComponentType.constantMapping	constantMapping
ParameterSwComponentType.dataTypeMapping	dataTypeMapping
RapidPrototypingScenario.rptContainer	shortName, variationPoint.shortLabel
RapidPrototypingScenario.rptSystem	rptSystem
RptContainer.byPassPoint	byPassPoint
RptContainer.rptContainer	shortName, variationPoint.shortLabel
RptContainer.rptHook	rptHook, variationPoint.shortLabel

<code>RTEEvent.disabledMode</code>	<code>contextPort</code> , <code>contextModeDeclarationGroupPrototype</code> , <code>targetModeDeclaration</code>
<code>SwcInternalBehavior.arTypedPerInstanceMemory</code>	<code>shortName</code> , <code>variationPoint.shortLabel</code>
<code>SwcInternalBehavior.event</code>	<code>shortName</code> , <code>variationPoint.shortLabel</code>
<code>SwcInternalBehavior.explicitInterRunnableVariable</code>	<code>shortName</code> , <code>variationPoint.shortLabel</code>
<code>SwcInternalBehavior.implicitInterRunnableVariable</code>	<code>shortName</code> , <code>variationPoint.shortLabel</code>
<code>SwcInternalBehavior.includedDataTypeSet</code>	<code>includedDataTypeSet</code>
<code>SwcInternalBehavior.includedModeDeclarationGroupSet</code>	<code>includedModeDeclarationGroupSet</code>
<code>SwcInternalBehavior.instantiationDataDefProps</code>	<code>instantiationDataDefProps</code> , <code>variationPoint.shortLabel</code>
<code>SwcInternalBehavior.perInstanceMemory</code>	<code>shortName</code> , <code>variationPoint.shortLabel</code>
<code>SwcInternalBehavior.perInstanceParameter</code>	<code>shortName</code> , <code>variationPoint.shortLabel</code>
<code>SwcInternalBehavior.portAPIOption</code>	<code>portAPIOption</code> , <code>variationPoint.shortLabel</code>
<code>SwcInternalBehavior.runnable</code>	<code>shortName</code> , <code>variationPoint.shortLabel</code>
<code>SwcInternalBehavior.serviceDependency</code>	<code>shortName</code> , <code>variationPoint.shortLabel</code>
<code>SwcInternalBehavior.sharedParameter</code>	<code>shortName</code> , <code>variationPoint.shortLabel</code>
<code>SwcInternalBehavior.variationPointProxy</code>	<code>shortName</code>
<code>SwComponentType.consistencyNeeds</code>	<code>shortName</code> , <code>variationPoint.shortLabel</code>
<code>SwComponentType.port</code>	<code>shortName</code> , <code>variationPoint.shortLabel</code>
<code>SwComponentType.swComponentDocumentation</code>	<code>swComponentDocumentation</code> , <code>variationPoint.shortLabel</code>
<code>SwcServiceDependency.assignedPort</code>	<code>assignedPort</code> , <code>variationPoint.shortLabel</code>

Table H.1: Usage of splittable elements

I 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 Table I.1 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 [12].

Variation Point	Latest Binding Time
ApplicationArrayElement.maxNumberOfElements	preCompileTime
ApplicationRecordDataType.element	preCompileTime
ArgumentDataPrototype.typeBlueprint	blueprintDerivationTime
ARPackage.arPackage	blueprintDerivationTime
ARPackage.element	systemDesignTime
ArrayValueSpecification.element	preCompileTime
AtomicSwComponentType.internalBehavior	preCompileTime
CalibrationParameterValueSet.calibrationParameterValue	preCompileTime
ClientServerInterface.operation	blueprintDerivationTime
ClientServerOperation.argument	blueprintDerivationTime
CompositionSwComponentType.component	postBuild
CompositionSwComponentType.connector	postBuild
CompositionSwComponentType.instantiationRTEEventProps	codeGenerationTime
CompuConstFormulaContent.vf	codeGenerationTime
CompuNominatorDenominator.v	preCompileTime
CompuScale.lowerLimit	preCompileTime
CompuScale.upperLimit	preCompileTime
CompuScales.compuScale	blueprintDerivationTime
ConsistencyNeeds.dpgDoesNotRequireCoherency	preCompileTime
ConsistencyNeeds.dpgRequiresCoherency	preCompileTime
ConsistencyNeeds.regDoesNotRequireStability	preCompileTime
ConsistencyNeeds.regRequiresStability	preCompileTime
DataPrototypeGroup.dataPrototypeGroup	preCompileTime
DataPrototypeGroup.implicitDataAccess	preCompileTime
EndToEndProtection.endToEndProtectionISignalIPdu	preCompileTime
EndToEndProtection.endToEndProtectionVariablePrototype	preCompileTime
EndToEndProtectionSet.endToEndProtection	preCompileTime
Implementation.buildActionManifest	codeGenerationTime
Implementation.generatedArtifact	preCompileTime
Implementation.requiredArtifact	preCompileTime
Implementation.requiredGeneratorTool	preCompileTime
ImplementationDataType.subElement	preCompileTime
ImplementationDataTypeElement.arraySize	preCompileTime
ImplementationDataTypeElement.subElement	preCompileTime
InternalBehavior.constantMemory	preCompileTime
InternalBehavior.exclusiveArea	preCompileTime
InternalBehavior.exclusiveAreaNestingOrder	preCompileTime
InternalBehavior.staticMemory	preCompileTime
InternalConstrs.lowerLimit	preCompileTime
InternalConstrs.upperLimit	preCompileTime
ModeDeclarationGroup.modeDeclaration	blueprintDerivationTime

NumericalOrText.vf	preCompileTime
NumericalValueSpecification.value	preCompileTime
NvBlockDescriptor.clientServerPort	preCompileTime
NvBlockDescriptor.instantiationDataDefProps	preCompileTime
NvBlockDescriptor.nvBlockDataMapping	preCompileTime
NvBlockSwComponentType.nvBlockDescriptor	preCompileTime
ParameterSwComponentType.instantiationDataDefProps	preCompileTime
PerInstanceMemorySize.size	preCompileTime
PhysConstrs.lowerLimit	preCompileTime
PhysConstrs.upperLimit	preCompileTime
PortGroup.outerPort	preCompileTime
PortInterfaceMappingSet.portInterfaceMapping	blueprintDerivationTime
RapidPrototypingScenario.rptContainer	preCompileTime
ReceiverComSpec.maxDeltaCounterInit	preCompileTime
ReceiverComSpec.usesEndToEndProtection	preCompileTime
RecordValueSpecification.field	preCompileTime
RptContainer.byPassPoint	preCompileTime
RptContainer.rptContainer	preCompileTime
RptContainer.rptHook	preCompileTime
RuleArguments.vf	preCompileTime
RuleArguments.vt f	preCompileTime
RuleBasedValueSpecification.arguments	preCompileTime
RunnableEntity.asynchronousServerCallResultPoint	preCompileTime
RunnableEntity.dataReadAccess	preCompileTime
RunnableEntity.dataReceivePointByArgument	preCompileTime
RunnableEntity.dataReceivePointByValue	preCompileTime
RunnableEntity.dataSendPoint	preCompileTime
RunnableEntity.dataWriteAccess	preCompileTime
RunnableEntity.externalTriggeringPoint	preCompileTime
RunnableEntity.internalTriggeringPoint	preCompileTime
RunnableEntity.modeAccessPoint	preCompileTime
RunnableEntity.modeSwitchPoint	preCompileTime
RunnableEntity.parameterAccess	preCompileTime
RunnableEntity.readLocalVariable	preCompileTime
RunnableEntity.serverCallPoint	preCompileTime
RunnableEntity.writtenLocalVariable	preCompileTime
RunnableEntityGroup.runnableEntity	preCompileTime
RunnableEntityGroup.runnableEntityGroup	preCompileTime
ScaleConstr.lowerLimit	preCompileTime
ScaleConstr.upperLimit	preCompileTime
SenderComSpec.usesEndToEndProtection	preCompileTime
ServiceDependency.assignedDataType	preCompileTime
SubElementMapping.firstElement	preCompileTime
SubElementMapping.secondElement	preCompileTime
SwAxisGeneric.swNumberOfAxisPoints	preCompileTime
SwAxisIndividual.swMaxAxisPoints	preCompileTime
SwAxisIndividual.swMinAxisPoints	preCompileTime
SwcBswMapping.runnableMapping	preCompileTime
SwcBswMapping.synchronizedModeGroup	preCompileTime
SwcBswMapping.synchronizedTrigger	preCompileTime
SwcImplementation.perInstanceMemorySize	preCompileTime
SwcInternalBehavior.arTypedPerInstanceMemory	preCompileTime
SwcInternalBehavior.event	preCompileTime

SwcInternalBehavior.explicitInterRunnableVariable	preCompileTime
SwcInternalBehavior.implicitInterRunnableVariable	preCompileTime
SwcInternalBehavior.instantiationDataDefProps	preCompileTime
SwcInternalBehavior.perInstanceMemory	preCompileTime
SwcInternalBehavior.perInstanceParameter	preCompileTime
SwcInternalBehavior.portAPIOption	preCompileTime
SwcInternalBehavior.runnable	preCompileTime
SwcInternalBehavior.serviceDependency	preCompileTime
SwcInternalBehavior.sharedParameter	preCompileTime
SwComponentDocumentation.chapter	postBuild
SwComponentType.consistencyNeeds	preCompileTime
SwComponentType.port	preCompileTime
SwComponentType.portGroup	preCompileTime
SwComponentType.swComponentDocumentation	preCompileTime
SwcServiceDependency.assignedData	preCompileTime
SwcServiceDependency.assignedPort	preCompileTime
SwGenericAxisParam.vf	preCompileTime
SwValues.vf	preCompileTime
SwValues.vtf	preCompileTime
TextTableMapping.bitfieldTextTableMaskFirst	preCompileTime
TextTableMapping.bitfieldTextTableMaskSecond	preCompileTime
TextTableValuePair.firstValue	preCompileTime
TextTableValuePair.secondValue	preCompileTime

Table I.1: Usage of variation points